CodeWarrior®

IDE User Guide

Because of last-minute changes to CodeWarrior, some of the information in this manual may be inaccurate. Please read the Release Notes on the CodeWarrior CD for the latest up-to-date information.

Revised: 980831–mds

Metrowerks CodeWarrior copyright ©1993–1998 by Metrowerks Inc. and its licensors. All rights reserved.

Documentation stored on the compact disk(s) may be printed by licensee for personal use. Except for the foregoing, no part of this documentation may be reproduced or transmitted in any form by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without permission in writing from Metrowerks Inc.

Metrowerks, the Metrowerks logo, CodeWarrior, and Software at Work are registered trademarks of Metrowerks Inc. PowerPlant and PowerPlant Constructor are trademarks of Metrowerks Inc.

All other trademarks and registered trademarks are the property of their respective owners.

ALL SOFTWARE AND DOCUMENTATION ON THE COMPACT DISK(S) ARE SUBJECT TO THE LICENSE AGREEMENT IN THE CD BOOKLET.

How to Contact Metrowerks:

U.S.A. and international	Metrowerks Corporation 9801 Metric Boulevard, Suite 100 Austin, TX 78758 U.S.A.
Canada	Metrowerks Inc. 1500 du College, Suite 300 Ville St-Laurent, QC Canada H4L 5G6
Ordering	Voice: (800) 377–5416 Fax: (512) 873–4901
World Wide Web	`http://www.metrowerks.com`
Registration information	`register@metrowerks.com`
Technical support	`support@metrowerks.com`
Sales, marketing, & licensing	`sales@metrowerks.com`
CompuServe	goto `Metrowerks`

Table of Contents

1 Introduction **15**

Introduction Overview 15
Metrowerks Year 2000 Compliance 17
Read the Release Notes! 17
Manual Conventions 18
Typographical Conventions 18
Host Conventions 19
Figure Conventions. 19
Keyboard Conventions 20
IDE User Guide Overview 21
About the CodeWarrior IDE 22
Where to Go From Here 23
QuickStart and Tutorials 24
Targeting Documentation 24
What's New in This Release 26
Editor Improvements 26
Project Manager Improvements 27
Integrated Debugger Improvements 27
Version Control Improvements. 27
Plug-in Diagnostics Support 27
Enhanced Online Help Support 28
Multiple-Monitor Debugging Support (Mac OS) 28

2 Getting Started **29**

Getting Started Overview 29
System Requirements 30
Windows Requirements. 30
Mac OS Requirements 30
Solaris Requirements 30
CodeWarrior IDE Installation. 31
Programming Concepts 31
Creating Input Files 32
Generating the Software. 33
Debugging and Refining 34

An Introduction to the CodeWarrior IDE 35
Projects and Targets 35
Source Code Editing and Browsing 36
Compiling and Linking 36
Project Debugging 37
Version Control 38
Customizing the IDE 38
External Tool Support (Mac OS) 38
Scripting (Mac OS) 38

3 Working with Projects 41

Working with Projects Overview 41
Guided Tour of the Project Window 42
Navigating the Project Window 43
Project Window Toolbar 43
File View 44
Link Order View 49
Targets View 50
Creating a New Project 50
Types of Project Files 51
Choosing the New Project Stationery File 51
Naming Your New Project 53
Modifying Your New Project 55
Building Your New Project 56
Working with Project Stationery 56
About Project Stationery 56
About the Project Stationery Folder 57
Creating Your Own Project Stationery 57
Opening an Existing Project 61
Using the Open Command 61
Using the Open Recent Command 62
Using the Project Window to Open Subprojects 62
Opening Project Files Created on Other Host Platforms . . . 63
Opening Project Files from Earlier IDE Versions 63
Saving a Project 65

Items Saved with Your Project 66
Saving a Copy of Your Project 66
Closing a Project 67
Choosing a Default Project 67
Managing Files in a Project 68
Expanding and Collapsing Groups 68
Selecting Files and Groups 70
Adding Files 71
Moving Files and Groups 79
Creating Groups 80
Removing Files and Groups 81
Renaming Groups 82
Touching and Untouching Files 83
Working with Complex Projects 85
What is a Build Target? 86
What is a Subproject? 86
Strategy for Creating Complex Projects 88
Creating a New Build Target 88
Changing a Target Name 90
Changing the Target Settings 90
Setting the Current Build Target 91
Creating Target Dependencies 91
Assigning Files to Targets 92
Creating Subprojects Within Projects 93
Examining Project Information 95
Moving a Project 96
Controlling Debugging in a Project 97
Activating Debugging for a Project 97
Activating Debugging for a File 97
Adding Preprocessor Symbols to a Project 99

4 Working with Files 101

Working with Files Overview 101
Creating a New File 101
Opening an Existing File 102

Opening Files from the File Menu 102
Opening Files from the Project Window 104
Opening Files from an Editor Window 107
Opening a Related File 108
Saving a File 108
Closing a File 115
Closing One File 115
Closing All Files 117
Printing a File 117
Setting Print Options 117
Printing a Window 118
Reverting to a Previously-Saved File 119
Comparing and Merging Files & Folders 120
File Comparison and Merge Overview 120
Choosing Files To Compare 122
Examining and Applying Differences 124
Choosing Folders To Compare 124

5 Editing Source Code 129

Source Code Editor Overview 129
Guided Tour of the Editor Window 130
Text Editing Area 131
Interface Pop-Up Menu 131
Routine Pop-Up Menu 132
Marker Pop-Up Menu 133
Options Pop-Up Menu 134
VCS Pop-Up Menu 135
File Path Caption 135
Dirty File Marker 136
Pane Splitter Controls 136
Line Number Button 136
Toolbar Disclosure Button 136
Path Pop-Up Menu (Mac OS) 137
Editor Window Configuration 137
Setting Text Size and Font 138

Seeing Window Controls 138
Splitting the Window into Panes 139
Saving Editor Window Settings 140
Basic Text Editing 141
Basic Editor Window Navigation 142
Adding Text 143
Deleting Text 144
Selecting Text 144
Moving Text (Drag and Drop) 146
Using Cut, Copy, Paste, and Clear 147
Balancing Punctuation 147
Shifting Text Left and Right 148
Undoing Changes 148
Controlling Color 150
Navigating the Text 150
Finding a Routine 151
Adding, Removing, and Selecting a Marker 151
Opening a Related File 153
Going to a Particular Line 154
Using Go Back and Go Forward 154
Configuring Editor Commands 155
Opening a File's Directory (Mac OS) 155
Online References 155
Finding Symbol Definitions 156
WinHelp (Windows) 157
QuickHelp (Mac OS) 158
QuickView (Mac OS) 158
THINK Reference (Mac OS) 159
Inserting Routine (Reference) Templates (Mac OS) 160
HyperHelp (Solaris) 160

6 Searching and Replacing Text 161

Searching and Replacing Text Overview 161
Guided Tour of the Find Dialog Box 161
Find and Replace Section 162

Multi-File Search Section 168
Searching for Selected Text 172
Finding and Replacing Text in a Single File 174
Finding Search Text 175
Controlling Search Range in a Single File 176
Controlling Search Parameters 177
Searching with Special Characters 178
Replacing Found Text 178
Using Batch Searches 180
Finding and Replacing Text in Multiple Files 181
Activating Multi-File Search 182
Choosing Files to be Searched 183
Saving a File Set 187
Removing a File Set 188
Controlling Search Range in Multiple Files 189
Using Regular Expressions (`grep`) 191
Special Operators 192
Using Regular Expressions 193

7 Browsing Source Code 201

Browser Overview 201
Activating the Browser 202
Understanding the Browser Strategy 202
Catalog View 204
Browser View 204
Hierarchy View 206
Guided Tour of the Browser 207
Context Pop-Up Menu 208
Catalog Window 209
Multi-Class Browser Window 211
Single-Class Browser Window 217
Multi-Class Hierarchy Window 219
Single-Class Hierarchy Window 221
Symbol Window 222
Using the Browser 225

Setting Browser Options 225
Identifying Symbols in the Browser Database 226
Navigating Code in the Browser 226
Browsing Across Subprojects. 228
Completing Symbols 229
Opening a Source File. 229
Seeing a Declaration 230
Seeing a Routine Definition 230
Editing Code in the Browser 231
Analyzing Inheritance 231
Finding Functions That Are Overrides 231
MFC Class Viewing (Windows) 232
PowerPlant Class Viewing (Mac OS) 232
Saving a Default Browser 233

8 Configuring IDE Options 235

Configuring IDE Overview 235
Preferences Guided Tour. 235
Preference Panels. 236
Dialog Box Buttons 236
Choosing Preferences 238
General Preferences. 238
Editor Preferences 260
Debugger Preferences. 270
Customizing Toolbars 280
Kinds of Toolbars. 280
Toolbar Elements. 281
Showing and Hiding a Toolbar. 282
Modifying a Toolbar 285
Restoring a Toolbar to Default Settings 287
Anchoring the Floating Toolbar (Mac OS) 287

9 Configuring Target Options 289

Configuring Target Options Overview. 289
Target Settings Guided Tour 290
Panels 291

Dialog Box Buttons 291
Choosing Target Settings 293
Target Configurations 295
Editor Configurations 312

10 Compiling and Linking 317

Compiling and Linking Overview 317
Choosing a Compiler 318
Understanding Plug-in Compilers 318
Setting a File Extension 319
Compiling and Linking a Project 319
Compiling Files 320
Setting Link Order 322
Updating a Project 322
Making a Project 323
Enabling Debugging 324
Running a Project 324
Debugging a Project 325
Generating a Link Map 326
Synchronizing Modification Dates 326
Removing Objects 327
Advanced Compile Options 328
Using Precompiled or Preprocessed Headers 329
Creating Precompiled Headers 330
Defining Symbols For C/C++ 333
Defining Symbols For Pascal 334
Preprocessing Source Code. 336
Disassembling Source Code 336
Guided Tour of the Message Window 337
Using the Message Window 340
Seeing Errors and Warnings 341
Stepping Through Messages 342
Correcting Compiler Errors and Warnings 343
Correcting Linker Errors 344
Correcting Pascal Circular References 347

Saving and Printing the Message Window 348
Locating Errors in Modified Files 349
Special Library Options (Mac OS) 350
Import Weak. 351
Initialize Before 351
Merge Into Output 351

11 Configuring Version Control Software 353
Version Control System Overview 353
Version Control System (VCS) Setup 354
Setting up a Project for Revision Control 356
Using Source Code Control with Files 359
Determining Version Control Status of a File 360
Modifying a Read-Only Source Code File 362
Other Revision Control Operations 363
Viewing a File's Status in the Project Window 365
Mac OS '`ckid`' Resources 365
Advanced VCS Operations. 366
Windows VCS Features 367
Mac OS VCS Features 369
VCS Window 372

12 IDE Menu Reference 373
IDE Menu Reference Overview 373
File Menu 374
Edit Menu 379
Search Menu 382
Project Menu 389
Debug Menu 395
Data Menu. 399
Window Menu 401
Version Control System (VCS) Menu 405
Help Menu 406
Windows Help Menu 407
Mac OS Help Menu. 407
Java Exceptions Submenu 408

Toolbar Submenu 408
Apple Menu (Mac OS) 410
Tools Menu (Mac OS) 410
Scripts Menu (Mac OS) 413
Editor Extensions Menu (Mac OS) 414
Info Menu (Solaris) 415

A Default CodeWarrior Key Bindings 417

Mac OS and Solaris Modifier Key Legend 417
File Menu 418
Edit Menu 420
Search Menu 421
Project Menu 422
Debug Menu 423
Data Menu 425
Window Menu 426
Misc. 428
Editor Commands 428
Prefix Keys 431

B Mac OS CodeWarrior Scripting 433

CodeWarrior Apple Events Overview 433
AppleScript Tools and Reference Material 434
Writing Your First CodeWarrior IDE AppleScript 435
CodeWarrior IDE AppleScript Events 436
Processing Errors 437
Required Events 439
File Handling Events 440
Building Events 446
Status/Query Events 455
Navigation Events 463
CodeWarrior IDE AppleScript Classes 465
Project Classes 466
Compiler Classes 477
CodeGen Classes 482
Disassembler Classes 486

Linker Classes 487
Build Classes 493
Browser Classes 495
Editor Classes 496
Object Classes 503
Misc Classes 506
Coding with CodeWarrior IDE and Apple Events 511

C Mac OS CodeWarrior and MPW ToolServer 513
Using MPW ToolServer Overview 513
About ToolServer 513
ToolServer and MPW Shell Differences 515
Installing ToolServer 515
Starting ToolServer 517
Executing MPW Commands 519
Using the ToolServer WorkSheet 520
Using the ToolServer Tools Menu 521
Using a Commando Dialog Box 523
Looking Up MPW 411 Documentation 528
Creating MPW Tool Projects 530

D Solaris Utilities 531
Overview 531
Copy Files Accessory 531
Source File/Directory Selection 532
Currently Selected Files/Directories 533
Destination Directory Selection 534
File Systems Facility 535
The File Systems Dialog Box 535
The File System Options Dialog Box 537
Keyboard Preferences Dialog Box 539

1

Introduction

This manual describes the CodeWarrior Integrated Development Environment (IDE) in detail. You can apply the adept capabilities of the IDE to develop software on different operating systems using programming languages like C, C++, and Pascal.

NOTE: On occasion a CodeWarrior product ships with an earlier version of the IDE than reflected in this user guide. In that case, your IDE will not have the new features described in this manual. You can identify new features by referring to "What's New in This Release" on page 26. Point your internet browser to the Metrowerks website at `http://www.metro-werks.com` for information on available patches and updates for CodeWarrior tools.

Introduction Overview

This section introduces the CodeWarrior IDE. The sections in this chapter are:

- Metrowerks Year 2000 Compliance
- Read the Release Notes!
- Manual Conventions
- IDE User Guide Overview
- About the CodeWarrior IDE
- Where to Go From Here
- What's New in This Release

The IDE is a collection of development tools for creating and generating code for the target systems listed in Table 1.1. To learn which manual to read for the target that you are interested in, refer to

Table 1.2. You use the same IDE when developing code for all of the following systems:

Table 1.1 CodeWarrior IDE Targets

Target	Description
BeOS	Be operating system
Embedded PowerPC	Various embedded PowerPC processors
Java	Java virtual machine
Mac OS	Mac OS running on either the 680x0 (68K) or PowerPC family of microprocessors
Nucleus	A real-time, preemptive, multi-tasking kernel for embedded systems
OS-9	A real-time operating system for software running on embedded systems
Palm OS	The operating system for the Palm OS connected organizer
PlayStation	The Sony PlayStation game console
Win32/x86	The Win32 model for Windows NT and Windows 95 on Intel 80x86-class and Pentium-class processors
Solaris	The Solaris operating system

NOTE: Your version of the CodeWarrior product does not contain compilers for all targets listed in Table 1.1. Check the product description of your CodeWarrior software for targets available to you.

The version of CodeWarrior you have supports some or all of these programming languages for developing applications:

- C and C++ — a compiler that implements templates, exception handling, run-time type information (RTTI), and inline assembly code.
- Java—for targeting Java virtual machines.
- Object Pascal—a compiler used for ANSI Pascal and Object Pascal, which provides support for Turbo Pascal Input/Output routines, conditional compilation and macros, extended debugging features, and inline assembly code.
- Assembly Language—a compiler supporting low-level machine code language.

For more information about these targets, see "Targeting Documentation" on page 24.

Metrowerks Year 2000 Compliance

The Products provided by Metrowerks under the License agreement process dates only to the extent that the Products use date data provided by the host or target operating system for date representations used in internal processes, such as file modifications. Any Year 2000 Compliance issues resulting from the operation of the Products are therefore necessarily subject to the Year 2000 Compliance of the relevant host or target operating system. Metrowerks directs you to the relevant statements of Microsoft Corporation, Sun Microsystems, Inc., Apple Computer, Inc., and other host or target operating systems relating to the Year 2000 Compliance of their operating systems. Except as expressly described above, the Products, in themselves, do not process date data and therefore do not implicate Year 2000 Compliance issues.

For additional information, visit: `http://www.metrowerks.com/about/y2k.html`.

Read the Release Notes!

Please read the release notes. They contain important information about new features, bug fixes, and incompatibilities that may not have made it into the documentation due to release deadlines. You

can find the release notes on the CodeWarrior CDs in the `Release Notes` folder.

Manual Conventions

The following sections describe the different conventions used in the CodeWarrior IDE manual:

- Typographical Conventions
- Host Conventions
- Figure Conventions
- Keyboard Conventions

Typographical Conventions

The IDE manual uses some style conventions to make it easier to read and find specific information:

Notes, warnings, tips, and beginner's hints

An advisory statement or **NOTE** may restate an important fact, or call your attention to a fact which may not be obvious.

A **WARNING** given in the text may call attention to something such as an operation that, if performed, could be irreversible, or flag a possible error that may occur.

A **TIP** can help you become more productive with the CodeWarrior IDE. Impress your friends with your knowledge of little-known facts that can only be learned by actually reading the fabulous manual!

A ***For Beginners*** note may help you better understand the terminology or concepts if you are new to programming.

Typeface conventions

If you see some text that appears in a `different` typeface (as the word `different` does in this sentence), you are reading file or folder names, source code, keyboard input, or programming items.

Text **formatted like this** means that the text refers to an item on the screen, such as a **menu command** or **control** in a dialog box.

If you are using an on-line viewing application that supports hypertext navigation, such as Adobe Acrobat, you can click on underlined and colored text to view another topic or related information. For example, clicking the text "IDE User Guide Overview" in Adobe Acrobat takes you to a section that gives you an overview of the entire IDE User Guide.

Host Conventions

CodeWarrior runs on the host platforms and operating systems listed below. Throughout this manual, a generic platform identifier is used to identify the host platform, regardless of operating system.

The specific versions of the operating system that host CodeWarrior are:

- **Windows**—desktop versions of the WIndows operating system that are Win32 compliant, such as Windows 95 or Windows NT.
- **Mac OS**—desktop versions of Mac OS, System 7.1 or later.
- **Solaris**—Solaris version 2.5.1 or later.

Figure Conventions

The visual interface of the hosts listed in "Host Conventions" on page 19 is nearly identical in all significant respects. When discussing a particular interface element such as a dialog box or window, the screenshot may come from any of these hosts. You should have no difficulty understanding the picture, even if you are using CodeWarrior on a different host than the one shown.

However, there are occasions when dialog boxes or windows are unique to a particular host. For example, a particular dialog box may appear dramatically different on a Windows host and on a Mac OS host. In that case, a screenshot from each unique host will appear and be clearly identified so that you can see how CodeWarrior works on your preferred host.

Keyboard Conventions

The default keyboard shortcuts for CodeWarrior on some platforms are very similar. However, keyboards and shortcuts do vary across host platforms. For example, a typical keyboard for a Windows machine has an Alt key, but that same key is called the Option key on a typical keyboard for a Mac OS computer.

To handle these kinds of situations, CodeWarrior documentation identifies and uses the following paired terms in the text:

- Enter/Return—the "carriage return" or "end of line" key. This is not the numeric keypad Enter key, although in almost all cases that works the same way.
- Backspace/Delete—the Windows Backspace key and the Mac OS Delete key. In most cases, CodeWarrior maps these keys the same way. This is the key that (in text editing) causes the character before the insertion point to be erased. (This is not the Delete/Del, the "forward delete" key.)
- Ctrl/Command—the Windows Ctrl (control) key and the Mac OS Command key (⌘). In most cases, CodeWarrior maps these keys the same way.
- Alt/Option—the Windows Alt key and the Mac OS Option key. In most cases, CodeWarrior maps these keys the same way.

For example, you may encounter instructions such as "Press Enter/Return to proceed," or "Alt/Option click the Function pop-up menu to see the functions in alphabetical order." Use the appropriate key as it is labeled on your keyboard.

Some combinations of key strokes require multiple modifier keys. In those cases, key combinations are shown connected with hyphens. For example, if you read "Shift-Alt/Option-Enter/Return,"

you would press the Shift, Alt, and Enter keys on a Windows host and the Shift, Option, and Return keys on a Mac OS host.

Sometimes the cross-platform variation in keyboard shortcuts is more complex. In those cases, you will see more detailed instructions on how to use a keyboard shortcut for your host platform. In all cases the host and shortcut will be clearly identified.

Special Note for Solaris Users

The Solaris-hosted CodeWarrior IDE uses the same modifier key names as used for Mac OS (Shift, Command, Option, and Control). Likewise, the Key Bindings preference panel uses Mac OS symbols to represent modifier keys. Table A.1 on page 418 shows the default modifier key mappings and the symbols used to represent them. On Solaris machines, modifier keys can be mapped to any key on the keyboard. See "Keyboard Preferences Dialog Box" on page 539 for more information on changing the default modifier key mappings. When reading this manual, you will need to keep in mind your modifier key mappings.

IDE User Guide Overview

There are several chapters in this User Guide to explain how to use the IDE. Each chapter begins with an overview of the topics discussed in that chapter. The chapter overviews are:

- Introduction Overview—(this chapter) an overview of the CodeWarrior IDE languages, targets, hosts, and documentation
- Getting Started Overview—system requirements, installation, guided tour of the user interface
- Working with Projects Overview—creating, configuring, and working with projects
- Working with Files Overview—opening, saving, backing up, comparing, and printing files
- Source Code Editor Overview—editing and navigating text and source code

- Searching and Replacing Text Overview—finding and replacing text
- Browser Overview—analyzing and navigating through a project from various views
- Configuring IDE Overview—customizing the IDE Preferences window and using toolbars
- Configuring Target Options Overview—setting up and customizing a project and its targets
- Compiling and Linking Overview—compiling, linking, running, updating, preprocessing, and precompiling a project's target and its files
- Version Control System Overview—using revision control systems with the CodeWarrior IDE
- IDE Menu Reference Overview—lists menus and menu commands
- Default CodeWarrior Key Bindings—lists all the default key bindings used by the CodeWarrior IDE
- CodeWarrior Apple Events Overview—lists Apple Event and AppleScript support provided by the CodeWarrior IDE
- Using MPW ToolServer Overview—using Apple's ToolServer with the CodeWarrior IDE

About the CodeWarrior IDE

The CodeWarrior IDE is an application that provides a simple, versatile graphical user interface and tools for developing computer software for many different platforms using different programming languages. Using the IDE, you can develop a program, plug-in, library, or other executable code to run on a wide variety of computer systems.

The CodeWarrior IDE permits a software developer to quickly assemble source code files (for example, a file written in the C++ computer language), resource files, library files, other project files, and configuration settings into a project, without writing a complicated build script (or "make file") for the project. Source code files may be

added or deleted from a project using simple mouse and keyboard operations instead of tediously editing a build script.

More than one configuration of files and settings may be added to a project. Such a configuration is called a build target. Build targets can share files in the project while using their own settings. For example, a project may contain a debugging build target which contains source code files, extra diagnostic source code files, and settings to specify the generation of debugger information. Another build target in the same project may use the some of the same files as the debugging build target, but uses settings that specify the use of compiler optimizations and no debugging information. After debugging a program, generating a final version is as simple as changing a project's build target and using a single Make command. For more information about build targets, see "Targeting Documentation" on page 24.

The CodeWarrior IDE includes compilers, linkers, a debugger, a source code browser, and an editor. These tools allow you to edit, navigate and examine your code, compile it, link it, and debug it until you have a running application. Options for code generation, debugging, and navigation of your project are all configurable in the IDE.

The CodeWarrior IDE and its tools have everything you need to develop code!

Where to Go From Here

If you are an experienced CodeWarrior IDE user, review "What's New in This Release" on page 26 for an overview of the new features.

When you are ready to debug, be sure to read the *CodeWarrior Debugger User Guide*.

If you are trying to get started quickly with a new platform or if you are new to CodeWarrior, see "QuickStart and Tutorials" on page 24.

To get started quickly with a new target operating system, see "Targeting Documentation" on page 24.

The following sections will give you a better idea of how to get help for your next step with CodeWarrior.

- QuickStart and Tutorials
- Targeting Documentation

QuickStart and Tutorials

You will find all the manuals mentioned in this section in the `CodeWarrior Documentation` folder on the CodeWarrior CD. For some products the manuals will be on the CodeWarrior Reference CD.

If you are new to the CodeWarrior IDE, check out the following resources:

- The *CodeWarrior QuickStart Guide* for an overview of CodeWarrior and pointers to the references available to you. The *QuickStart Guide* is on your CD, so you can use it regardless of whether you purchased any printed documentation.
- "An Introduction to the CodeWarrior IDE" on page 35 provides a quick overview of the CodeWarrior IDE user interface.
- The CW Core Tutorials folder on the CodeWarrior CD contains some sample projects that will help you become productive quickly with the CodeWarrior IDE.
- The instructions for using the viewers Metrowerks provides for on-line documentation in the CodeWarrior Documentation folder on the CodeWarrior CD.

Targeting Documentation

This manual describes how to use the CodeWarrior IDE. However, the manual doesn't show you how to use CodeWarrior to develop software for a specific "target."

The term "target" has two distinct meanings in this documentation and in CodeWarrior terminology generally. These meanings are:

- platform target—the particular operating system or microprocessor for which you are writing code
- build target—the particular collection of settings and files that determines what is in your code, and how your code is compiled and liked into the final output

For example, if you choose to write code for a particular processor running a desktop OS, you are specifying a platform target.

On the other hand, if you choose to compile your code with certain language features enabled, then you are changing the settings in the build target. If you choose to add or remove a file from a particular build, you are also changing the build target.

For those familiar with the term "makefile," a build target is analogous to a makefile.

For those already familiar with CodeWarrior, the Project window's Targets view is the collection of build targets in the project. One option within each build target is the choice of linker (in the Target Settings panel). The choice of linker specifies the platform target.

The context within which the term "target" appears will usually tell you whether it refers to a platform target or a build target. In those cases where ambiguity might occur, the documentation will use the full terminology, "platform target" or "build target," as appropriate.

Table 1.2 identifies the documentation that you should consult when developing software for specific platform targets.

NOTE: The CodeWarrior product you're using may not generate software for all the targets listed in Table 1.2.

Table 1.2 Targeting Guides for Various CodeWarrior Targets

Target	Targeting Manual
Java Virtual Machine	*Targeting the Java VM*
Mac OS	*Targeting Mac OS*

Target	Targeting Manual
MIPS	*Targeting MIPS*
Nucleus	*Targeting Nucleus*
Palm OS	*Targeting Palm OS*
PlayStation	*Targeting PlayStation OS*
PowerPC Embedded	*Targeting PowerPC for Embedded Systems*
Solaris	*Targeting Solaris*
Win32/x86	*Targeting Win32*

What's New in This Release

The CodeWarrior IDE is now at version 3.2. Among its new features, the following are the most notable:

- Editor Improvements
- Project Manager Improvements
- Integrated Debugger Improvements
- Version Control Improvements
- Plug-in Diagnostics Support
- Enhanced Online Help Support
- Multiple-Monitor Debugging Support (Mac OS)

Editor Improvements

The CodeWarrior IDE Editor features several subtle improvements to make your code editing tasks more efficient.

The Windows version of the CodeWarrior IDE now supports an external editor for displaying text files. Command lines customize the CodeWarrior IDE for use with the external editor. For more information, refer to "Use Third Party Editor (Windows)" on page 244.

The Mac OS version of the CodeWarrior IDE now supports the Appearance Manager and Navigation Services.

Project Manager Improvements

The CodeWarrior IDE's performance has been greatly improved when handling large projects. You will notice this performance gain when opening a project, searching for files, adding files to a project, compiling a project, and building a project over a network.

The IDE will now automatically create the necessary directories when the output folder specified in the Target Settings preference panel does not exist.

Project data folders have changed in order to accommodate the IDE's support of external object files. Each target now has a separate subfolder in the Data folder. Older projects are automatically re-organized to this new format when opened in the 3.2 IDE.

Integrated Debugger Improvements

The integrated debugger now features a "Windowing" preference panel. Using this preference panel, you can reduce screen clutter and organize windows during debugging sessions.

New contextual menus provide convenient access to commonly-used commands, such as formatting and setting watchpoints.

For more information, see "Windowing Panel" on page 272.

Version Control Improvements

A new Version Control Progress window shows the current version control status on a file and occasionally presents information about completed version control actions.

Plug-in Diagnostics Support

The IDE now includes a diagnostic feature that helps you troubleshoot plug-in development for CodeWarrior. You can access infor-

mation about the IDE's problems with loading the plug-in, as well as the properties of the installed plug-in.

For more information, read "Plugin Diagnostics" on page 240.

Enhanced Online Help Support

A new help system can guide you in using the CodeWarrior IDE. This system uses the WinHelp (Windows) and QuickHelp (Mac OS) applications to provide context-sensitive help. Reference material on performing single tasks, as well as series of tasks, is available.

For more information, see "Online References" on page 155.

Multiple-Monitor Debugging Support (Mac OS)

The Mac OS version of the integrated debugger now supports multiple-monitor debugging. Options are available for moving debugging windows to another monitor at the start of a debugging session and for opening debugging windows on the alternate monitor while a debugging session is in progress.

For more information, see "Multi-Monitor Debugging (Mac OS)" on page 274.

2

Getting Started

This chapter helps you get started using the CodeWarrior IDE (Integrated Development Environment). In it you'll find the system requirements and information about installing IDE software and an introduction to the IDE's user interface and capabilities.

Getting Started Overview

This chapter shows you how to get started using the CodeWarrior IDE.

The sections in this chapter are:

- System Requirements
- CodeWarrior IDE Installation
- Programming Concepts
- An Introduction to the CodeWarrior IDE

NOTE: This manual describes how to use the CodeWarrior IDE; it doesn't show you how to use CodeWarrior to develop software for a particular processor or operating system "target." For information on developing software for a target, see "Targeting Documentation" on page 24.

TIP: For a quick look at the IDE's features, see "An Introduction to the CodeWarrior IDE" on page 35. The tour gives you your first glimpse of the CodeWarrior IDE.

System Requirements

The system requirements to operate the CodeWarrior IDE are specified by platform. These platforms include:

- Windows Requirements
- Mac OS Requirements
- Solaris Requirements

Windows Requirements

The Windows-hosted version of CodeWarrior requires a 80486DX or Pentium™-class processor, 24 megabytes of RAM, Microsoft Windows 95 or Windows NT 4.0 operating system or later versions of these operating systems, and a CD-ROM drive to install the software.

For the best performance, we recommend you use a computer equipped with a Pentium™-class processor with at least 24 megabytes of RAM.

Mac OS Requirements

The Mac OS-hosted version of the CodeWarrior IDE requires a Motorola MC68040 or greater processor, or PowerPC 601 or greater processor, 24 megabytes of free RAM, Color QuickDraw, Mac OS System 7.5 or later, and a CD-ROM drive to install the software. When running on 68k machines, CFM 68K Enabler 4.0 is required (provided by the installer).

For the best performance, we recommend you use a Mac OS computer equipped PowerPC microprocessor, with at least 32 megabytes of free RAM.

Solaris Requirements

The Solaris-hosted version of the CodeWarrior IDE requires a Sun SparcStation or Sparc-based machine, at least 32 MB of RAM, a CD-ROM drive to install the software, 40 MB of free hard disk space,

Network Information Service, an X11 server (Open Windows v3.3 recommended), a window manager that is vX11r5 or later, and Motif 1.2.2 or later.

CodeWarrior IDE Installation

To learn how to install the CodeWarrior IDE, read the QuickStart guide. The CodeWarrior installation software places the CodeWarrior IDE, compilers, linkers, tools, and debuggers on your hard drive.

Programming Concepts

If you are new to programming computers, read this section to learn about the terms used in this manual.

There are three important tasks involved in developing software:

- Creating Input Files
- Generating the Software
- Debugging and Refining

First, you create files that contain statements using a computer language such as C, C++, Java, Pascal, or assembly language. You can also create files that contain resources, which are descriptions of user interface objects such as windows, dialog boxes, and menus.

Next, you apply tools to help turn your "source" materials into an output file that executes on your target computer. These tools include compilers, linkers, other tools, and maybe even assembly-language assemblers. You'll use these tools to create many different types of software, depending on the tools used and the target computer. Some of the types of software you can create include applications (or *executables*), dynamic (shared) libraries, and static libraries.

Once you have created a piece of software, you can use a debugger to examine the software. If the software does not run properly, you return to the first step to make changes to your source material so that the software behaves correctly.

Creating Input Files

The types of files you create and use to make a program are source code files, resource files, interface or header files, library files, and project files.

Source code file

A source code file is a text file containing program statements written in a language such as C, C++, Java, Pascal, or assembly language.

Resource file

A resource file contains descriptions of user interface items, such as window definitions, dialog box layouts, and text strings. A resource file may be a binary file linked into your software product, or it may be a text file translated by a special resource compiler before being linked. Placing resources in separate files makes it convenient to tweak and customize these items without having to recompile other parts of a program.

Interface or header file

An interface file, also called a header or include file, is a text file referred to by source code files. Typically, these files give access to objects, variables, data structures, routines, and other items in libraries or source code files.

Library file

A library file contains objects, variables, routines, and other items that have already been compiled. There are two kinds of libraries: static and dynamic.

The IDE builds a static library right into a program. A static library cannot change (hence the term “static”) unless you rebuild the program. Static libraries can’t be shared with other programs.

Although a program might refer to a dynamic library, the IDE doesn’t build it into a program. Instead, the program hooks up with the library dynamically, that is, while the program is actually run-

ning. Often, more than one program shares the same dynamic library. Also, a dynamic library may be replaced with a newer library without affecting the programs that refer to it as long as the newer library provides the same operations as its predecessor.

NOTE: Most processor and operating system targets support static libraries, but not all targets support dynamic libraries. For information on creating and using libraries for a specific target, see "Targeting Documentation" on page 24.

Project file

A project file is a file that contains one or more targets. Each target contains a list of source code, resource, interface, library files, and even other projects files that are used to create your software. Each target also contains its own settings that instruct the IDE how your software should be created. For more information about these targets, see "Targeting Documentation" on page 24.

Generating the Software

The types of tools the IDE uses to build your software include compilers, assemblers, and linkers. Using information from files and settings in a project's target, the IDE automatically choose the appropriate tools to create your software.

Compilers

A compiler translates a source code file, such as a C, C++, Pascal, or Java file into binary machine code (also called object code) that will be used by linkers in a later build stage. A compiler is one of the first tools that the IDE invokes to build your program.

Assemblers

An assembler translates an assembly language source code file into object code. An assembler is really just a compiler, only it translates assembly language source code rather than high-level language source code.

Linkers

A linker combines the object code in your project's target produced by the compilers and assemblers into a piece of software.

Output file

The output file is the piece of software that the linker generates. There are many different types of software, such as applications and libraries. For in-depth information about the kinds of software to develop for a target, see the *CodeWarrior Targeting* manual appropriate for your platform. For information on which targeting manual to read, see "Targeting Guides for Various CodeWarrior Targets" on page 25.

Debugging and Refining

A debugger is a piece of software that facilitates controlled program execution and enables developers to find errors in code. With it, you can stop at points in your program's execution and see variable contents. You can also execute one line of code at a time. The debugger thus helps you pinpoint problem areas in your code.

All you need to do to enable the debugger is produce special data that the debugger uses to control your program. The IDE creates *debugging information* or *symbolics* files, which contain the information the debugger needs to display and control statements, variables, objects, and data types in your source code.

The IDE includes an integrated debugger that makes the debugging process even more transparent. In addition, support has been added for third-party debuggers on some platforms. See the Targeting manual associated with your platform for additional information.

For information on using the integrated debugger, see the *CodeWarrior Debugger User Guide*.

An Introduction to the CodeWarrior IDE

This section describes the tasks and operations you do with the CodeWarrior IDE. The topics in this section are:

- Projects and Targets
- Source Code Editing and Browsing
- Compiling and Linking
- Project Debugging
- Version Control
- Customizing the IDE
- External Tool Support (Mac OS)
- Scripting (Mac OS)

For information on the terms used in this section, see "Programming Concepts" on page 31.

Projects and Targets

The IDE uses projects and build targets to organize the files and settings used to create a program. A project is a file that contains one or more targets. A build target is a collection of source code files, resource files, libraries, even other projects, and settings that describe how to create a piece of software for a particular processor or operating system. Build targets within a project may share the same files, but each target has its own settings.

The CodeWarrior IDE also has pre-configured stationery projects. Creating a new project is as easy as deciding on a target platform to develop for and a programming language to use, then choosing the corresponding stationery project.

Through a build target's settings dialog box, you may set options to choose the target processor or operating system to generate software for, customize compiler optimizations and other object code details, customize source code translation, specify the kinds of files that may be added to a target, and configure other settings.

For more information on working with projects and build targets, see "Targeting Documentation" on page 24, "Working with Projects Overview" on page 41, and "Configuring Target Options Overview" on page 289.

Source Code Editing and Browsing

The IDE has a powerful, flexible text editor for editing source code and text files. Besides regular text editing features, the IDE also provides drag-and-drop editing, the ability to open, edit, and save text files in Mac OS, Unix, and MS-DOS formats, and setting and jumping to markers in a text file.

The editor also has many advanced features for programmers. Some of its programming aids are: auto-indenting, syntax highlighting, routine and interface pop-up menus, and automatic balancing for braces, brackets, and parentheses. The editor works closely with the class browser to make editing, viewing, and navigating among routines, data structures, variables, and objects intuitive and quick.

The Find dialog box and the Search Menu have commands and features to search and replace text in one file or in a group of files. Text to search for and replace may be normal text or regular expressions.

A file comparison and merging command displays two text files side-by-side and lets you easily compare differences between the two files. In addition, you can compare the contents of two folders to look for differences between files.

For more information on working with text and source code files, see "Working with Files Overview" on page 101, "Source Code Editor Overview" on page 129, "Searching and Replacing Text Overview" on page 161, "Browser Overview" on page 201.

Compiling and Linking

The IDE has commands to preprocess, precompile, compile, update, link, run, run from a debugger, disassemble, and check syntax. The IDE automatically chooses the appropriate compilers and linkers to use and automatically determines which files to operate on when

you issue a compile or link command. The IDE uses the settings in a project's target to instruct its compilers and linkers how to process a target's files and data. All you have to do is use one of the commands in the Project Menu.

For more information on setting options, compiling, linking, and other software generation operations, see "Configuring Target Options Overview" on page 289 and "Compiling and Linking Overview" on page 317.

Project Debugging

The IDE features an integrated debugger to provide a seamless interaction between the programming and debugging of your source code. Some of the benefits that you will experience include:

- Reduced memory requirements. With only one application running, memory demands are significantly reduced.
- Increased productivity. Since you don't have to switch back and forth between the IDE and Debugger to step through your code, set breakpoints, etc., a more efficient use of time is achieved, increasing your productivity.
- The integrated debugger fully supports x86, PowerPC, 68K, and Java debugging. Separate debuggers are no longer required to debug each platform. The integrated debugger handles them all.

Once you enable the integrated debugger, you simply **Run** the project to activate debugging. You can pause the program at any time to set breakpoints, view variables or memory, step into or out of routines, as well as perform many other debugging tasks.

NOTE: Some versions of the CodeWarrior IDE do not ship with the integrated debugger. In those cases, debugging support is provided by the external MW Debug application or other third-party debuggers. See your platform's Targeting manual for additional information on debugging a specific target.

Version Control

The IDE can be configured to work with Version Control Systems (VCS). The IDE will log onto a file server, retrieve files, store files, and do other revision control tasks with a VCS software package that supports the CodeWarrior IDE.

For information on using the IDE with VCS software, see "VCS Pop-Up Menu" on page 135 and "Version Control System Overview" on page 353.

Customizing the IDE

The IDE has many user-configurable options. Use the IDE's Preferences dialog box to customize features like set the colors and fonts used to view and edit source code and assign keyboard shortcuts for commands. The IDE also has conveniently-placed, easily customizable toolbars that give quick access to commands and information by simply clicking a button.

For more information on customizing the IDE, see "Configuring IDE Overview" on page 235.

External Tool Support (Mac OS)

The IDE works with other tools like third-party text editors, debuggers, and Apple's ToolServer on the Mac OS.

For information on using CodeWarrior with other software development tools, see "IDE Extras Panel" on page 242, "Debugging a Project" on page 325, "Version Control System Overview" on page 353, and "Using MPW ToolServer Overview" on page 513.

Scripting (Mac OS)

The IDE supports Apple Events and AppleScripting on Mac OS. Repetitive, time-consuming, or complex tasks can be automated through AppleScript scripts. Use the Scripts Menu (Mac OS) to list and run scripts for the IDE.

For more information on scripting the IDE, see "CodeWarrior Apple Events Overview" on page 433 and "Scripts Menu (Mac OS)" on page 413.

3

Working with Projects

This chapter introduces the CodeWarrior IDE project window, and shows how to create, configure, and work with projects.

Working with Projects Overview

A project contains one or more build targets. Each build target in a project contains a collection of files that the IDE uses to build an output file. Build targets within a project may share some or all of their files. Some examples of an output file include an application, static library, or dynamic library.

Each build target within a project has its own options that customize how the IDE builds the target's output file. There are a wide variety of options that control code optimization, the browser, compiler warnings, and much more.

Finally, build targets within a project can be configured to depend on other targets in the project. This feature makes it possible to build software that, for example, combines the output files for different target processors into a single output file.

This chapter discusses many of the basic tasks involving projects, such as creating, opening, adding files, and saving projects. It also describes advanced operations such as moving files in the project window, marking files for debugging, creating nested projects and targets, and dividing the project window into segments or groups of files.

The topics in this chapter are:

- Guided Tour of the Project Window
- Creating a New Project
- Working with Project Stationery
- Opening an Existing Project
- Saving a Project
- Closing a Project
- Choosing a Default Project
- Managing Files in a Project
- Working with Complex Projects
- Project Inspector window for targets
- Moving a Project
- Controlling Debugging in a Project
- Debug Info markers

Guided Tour of the Project Window

The Project window shows information about the files and targets in a project file in three different views: the File view, Link Order view (sometimes called Segments), and Targets view. To choose a view, click its tab at the top of the Project window, as shown in Figure 3.1.

Figure 3.1 The view tabs at the top of the Project window

The File view shows a list of all the files in a project. Items in this view may be organized into hierarchical groups that you create and arrange.

The Link Order view (Segments) shows information about how the IDE will compile or link the final output file for the project's current target.

The Targets view shows information about the active target, target dependencies, and link-compatible targets.

The topics that explain the Project window in detail are:

- Navigating the Project Window
- Project Window Toolbar
- File View
- Link Order View
- Targets View

To learn more about debugging information, see "Controlling Debugging in a Project" on page 97.

Navigating the Project Window

To navigate the Project window, use the vertical scroll bar on the right side of the window, or the Up and Down Arrow keys on your keyboard. If your project window contains many files, use the Home and End keys to jump from the first file in the first segment or group (Home Key) to the last file in the last segment or group (End Key).

Use the Page Up and Page Down keys to scroll your project window one page up or one page down.

To learn about a technique for selecting files as you type, refer to "Selection by keyboard" on page 71.

Project Window Toolbar

The toolbar in the Project window has buttons and other items to provide shortcuts to commands and information about the project. You may configure which items are on the toolbar, and the order in which they are displayed. You can even choose to hide or display the toolbar. To learn more about toolbars in the CodeWarrior IDE

and how to configure them, refer to "Customizing Toolbars" on page 280.

File View

The Project window's **File** view shows the files for all targets in the project. The files can be arranged hierarchically into groups without affecting the way the IDE builds a target. This view also shows information about file access paths, data size, code size, debugging, target, modification status, and other information.

The topics describe the parts of the File view. These topics are:

- File column
- Code column
- Data column
- Debug column
- Target column
- Touch column
- Interface pop-up
- File Control pop-up
- Checkout Status column
- Project Checkout Status icon

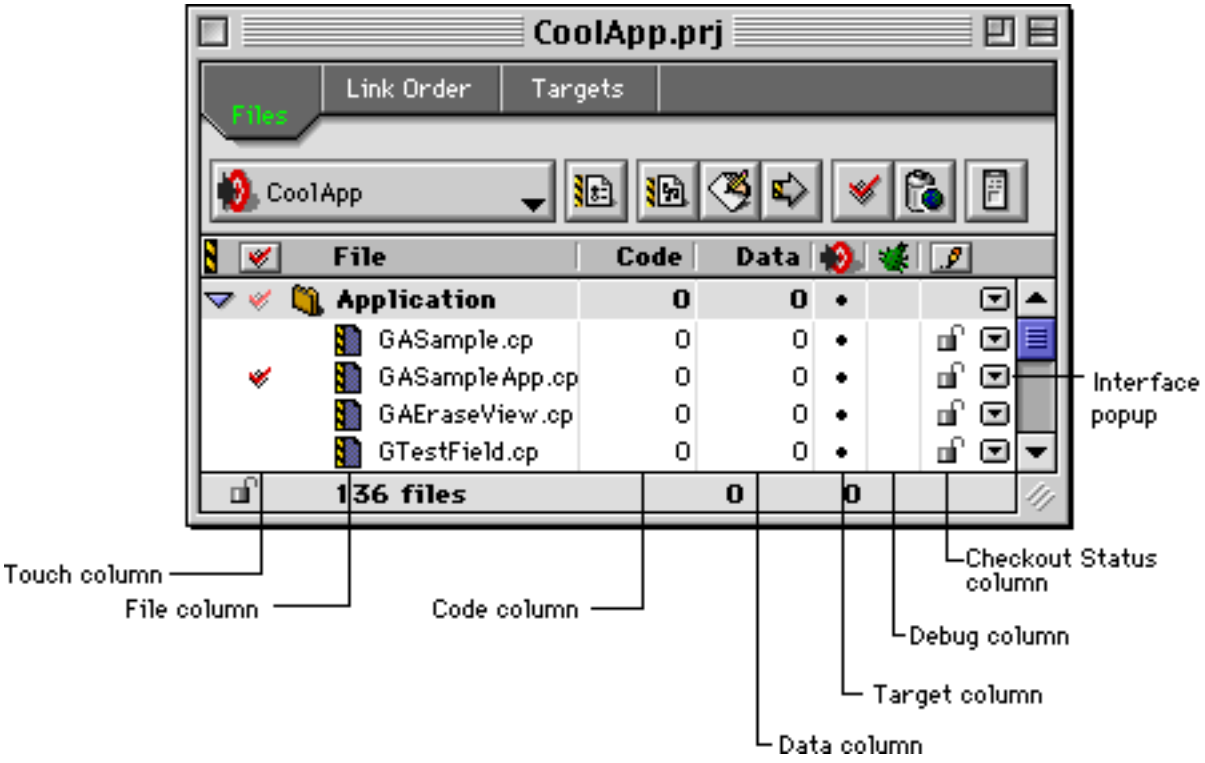
File column

Lists files in the project in a user-configurable hierarchical view. The file column lists the files and groups in a project. A group may contain files and other groups.

Double-clicking a file's name in the File column will open the file. For information on opening files from the File Column, refer to "Opening Files from the Project Window" on page 104.

Use the expand and collapse controls to view and hide the contents of groups in the File column. These controls are called tree controls in Windows and disclosure triangles in Mac OS.

Figure 3.2 The Project window



Code column

Shows the size, in bytes or kilobytes, of the compiled executable object code for files and groups. If zero is displayed, it means that your file has not yet been compiled. If "n/a" is displayed in a file's Code column, the file has no executable code that was compiled by the IDE.

The values in this column do not reflect the amount of object code that will be added to the final output file. The linker may not use all of a file's object code when creating the final output. Instead, the linker may ignore data and code that are not referenced by other files in the project ("dead stripping").

For more information on how the linker works, see "Compiling and Linking a Project" on page 319.

Data column

Shows the size, in bytes or kilobytes, of non-executable data in the object code for files in the project. A zero means that your file has not yet been compiled, or that the file does not contain a data section in its object code. If "n/a" is displayed in a file's Data column, the file has no object code data.

As in the Code column, the data values do not reflect the amount of data that will be added to the final output file.

The numeric values shown are only for items in the current target. Values for items not in the current target are shown in gray.

For more information on how the linker works, see "Compiling and Linking a Project" on page 319.

Debug column

Indicates that debugging information will be generated for the files in the project. A black marker in this column means that the IDE will generate debugging information for the corresponding item. A gray marker indicates that the IDE will generate debugging information for only some of the files in the group.

To generate debugging info for a:

- **File**—click in the Debug column next to the file.
- **Group**—click in the Debug column next to the group.
- **Project**—Shift/Option click in the Debug column.

For more information about debugging information for a file, see "Activating Debugging for a File" on page 97.

Target column

Indicates whether or not an item is in the project's current build target. The IDE displays this column if a project has more than one build target. A dark marker in this column means that the corresponding item is in the current build target. A gray marker indicates that only some of the files in that group are in the current build target.

To assign or unassign a current target for a:

- **File—**click in the Target column next to the file.
- **Group—**click in the Target column next to the group.
- **Project**—Shift/Option click in the Target column.

For information on adding or removing a file to or from a target using the Target column, see "Assigning Files to Targets" on page 92.

Touch column

Indicates whether a file needs to be compiled the next time a target is built. A marker in this column means that the corresponding item will be re-compiled for the next Bring Up To Date, Make, Run, or Debug command. A gray marker indicates that only some of the files in that group are set for re-compiling.

To indicate re-compiling or disable re-compiling for a:

- **File—**click in the Touch column next to the file.
- **Group—**click in the Touch column next to the group.
- **Project—**Shift/Option click in the Touch column.

To learn more about this feature, see "Synchronizing modification dates" on page 84.

Interface pop-up

Lists and opens interface and header files for your project source files. The Interface File Pop-up also allows you to touch or untouch its corresponding item and set other options that appear depending on the current target.

For groups, this pop-up menu lists the files within the group. Choosing a file in the popup will open the file.

For more information about opening interface and header files, see "Interfaces pop-up menu" on page 106.

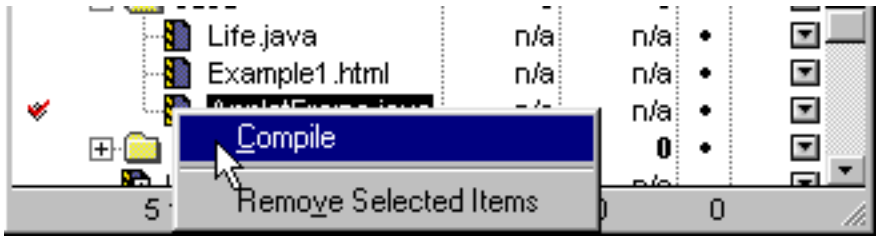
File Control pop-up

Right-click (Windows), Control-click (Mac OS), or click and hold (Solaris) on an item's icon in the Project window to view the File Control pop-up menu, as shown in Figure 3.3. From this menu, choose a command to operate on the item.

Windows To see the path along which the IDE accesses a file, select **Open in Windows Explorer** from the pop-up menu.

Mac OS To see the path along which the IDE accesses a file, highlight **File Path** in the pop-up menu. To view the file and its enclosing folder from the Finder, select **Reveal in Finder** from the pop-up menu.

Figure 3.3 File Control pop-up menu in the project window



Checkout Status column

Check files in or out of a version control system (VCS). Use this column to track changes to your code, particularly when more than one person is working on your software project.The Checkout Status Column, shown in Figure 3.2 on page 45, only appears when you have configured your CodeWarrior project to use a source code revision control system, such as *Metrowerks Visual SourceSafe for Macintosh* (formerly Metrowerks CodeManager) or *Microsoft Visual SourceSafe* (for Microsoft Windows).

To learn more about using your particular revision control system, refer to its documentation.

Project Checkout Status icon

Shows whether the project is writable or not, or what file access permissions are set for the project. A revision control system, such as

Metrowerks Visual SourceSafe for Macintosh or *Microsoft Visual SourceSafe* (for Microsoft Windows), assigns these permissions when you check a project file in or out of the revision control system.

To learn more about what the Project Checkout Status icon means, and how the icon relates to the access permissions assigned to your source code files, refer to "Determining Version Control Status of a File" on page 360.

To learn more about using your particular revision control system, refer to its documentation.

Link Order View

By default, the CodeWarrior IDE compiles files in the order that appears in the File view, but will fail if one file depends on information from a second file that has not been compiled yet. By putting the files in a correct order in the Link Order view the user can avoid this problem.

Items in the Link Order view can only be nested one level deep. The files in this view are arranged in the order they will be compiled. Changing their order affects the final binary code that is produced by your project file.

For more information, see "Setting Link Order" on page 322.

For more discussion about groups, refer to "Managing Files in a Project" on page 68.

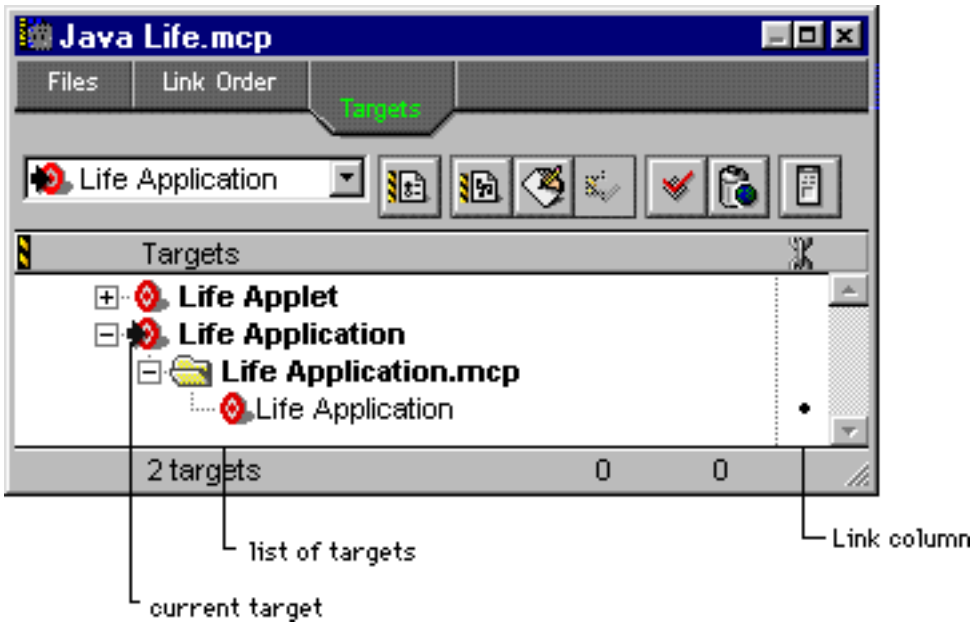
Mac OS The Link Order view may be named Segments for some targets. Refer to "Targeting Documentation" on page 24 for more information.

Embedded The Link Order view may be named Overlays for some targets. Refer to "Targeting Documentation" on page 24 for more information.

Targets View

The Targets view shows a list of the build targets in the project. This view also shows the objects that the build targets depend on to create a final output file. Figure 3.4 shows an example Targets view.

Figure 3.4 The Targets view in the Project window



For information on working with build targets, see "Working with Complex Projects" on page 85.

Creating a New Project

There are a few short steps involved in creating a new project:

- Types of Project Files
- Choosing the New Project Stationery File
- Naming Your New Project
- Modifying Your New Project
- Building Your New Project

Types of Project Files

CodeWarrior uses two types of files to create new projects. These two types are:

- Project Stationery—Project stationery contains pre-configured libraries, source code placeholders, and resource file placeholders. This kind of project file is useful for quickly creating new projects. Novice programmers and programmers that want the convenience of pre-configured settings should use project stationery.
- Empty Project—Empty project files do not contain any files whatsoever. This kind of project file is useful for custom configuration of compiler and linker settings. Advanced users can take advantage of empty projects to create fully-customized applications.

The rest of this chapter discusses creating new projects using project stationery.

NOTE: Advanced users who are interested in using empty projects should read "Configuring Target Options" on page 289. That chapter provides information about configuring compiler and linker settings for use with projects.

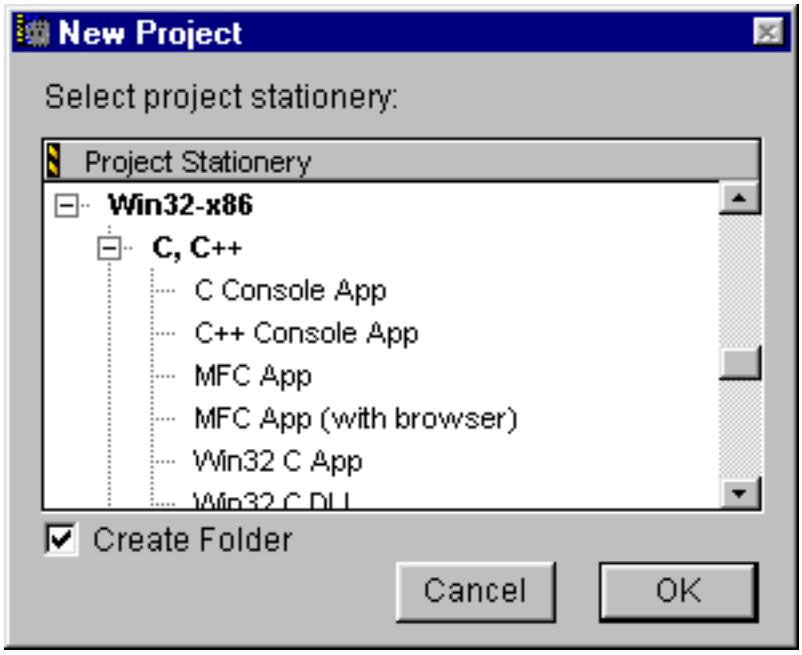
Choosing the New Project Stationery File

To create a new project, select the **New Project** command from the File Menu. The CodeWarrior IDE displays the New Project dialog box, shown in Figure 3.5. Use this dialog box to choose your project stationery.

All of the items in the New Project dialog box are arranged in a hierarchy. The first items you see are the various types of available stationery. You can choose from empty project stationery or project stationery for various operating systems and languages. As you progress through the hierarchy, you can select a specific operating system, language, and programming framework. When you have

chosen these more general options, you will be able to choose from several pre-configured projects.

Figure 3.5 New Project dialog box



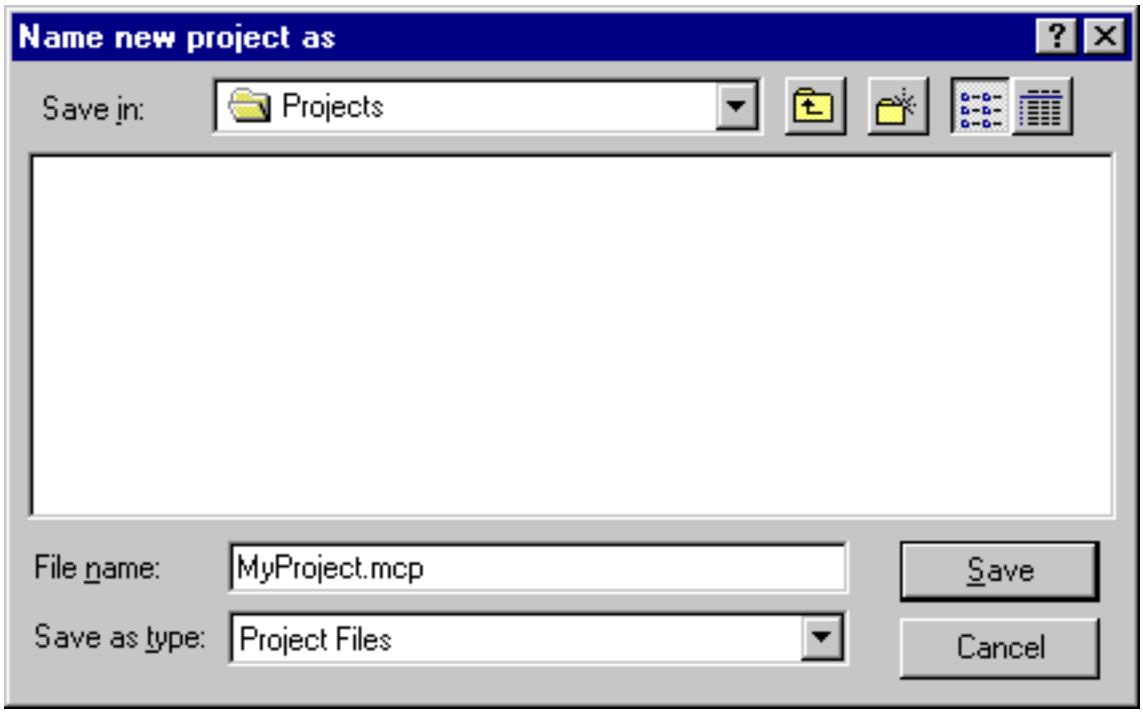
For example, suppose you want to create a new project that is geared towards creating a C++ console application for the Win32 platform. First, you would click on the tree control (Windows) or disclosure triangle (Mac OS) next to the "Win32-x86" option. This displays the languages in the Win32 platform for which stationery is available. Next, you would click the control next to the "C, C++" option. This shows the project stationery for the C and C++ programming languages. When you select "C++ Console App" and click **OK**, the CodeWarrior IDE creates a new project with all the libraries and support files you need. The "C Console App" project stationery provides a simple console-based application structure suitable for small non-frameworked programming examples.

To create an empty project that contains no libraries or other support files, you may choose the **Empty Project** option from the list.

For beginners: Choosing **Empty Project** is not recommended because of the complexities of including the correct libraries and files and choosing the correct target settings. Instead of creating an empty project, use the project stationery which is included with CodeWarrior. These stationery projects are pre-configured and convenient to use.

To create a new folder to contain the project file and all of its associated files, make sure the **Create Folder** checkbox is selected.

Figure 3.6 Naming a new project (Windows)



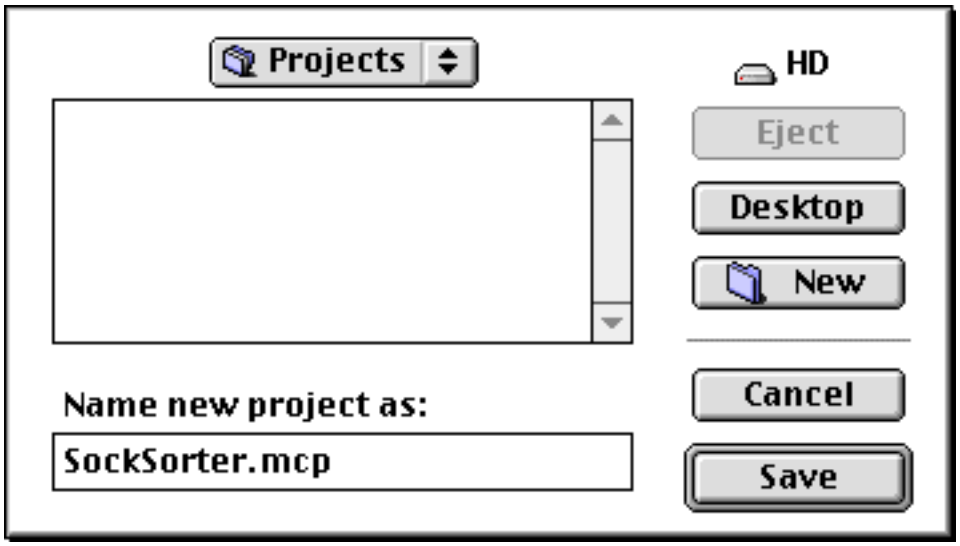
Naming Your New Project

After choosing **New Project** from the File Menu and selecting an item in the New Project window, click the **OK** button and a dialog box appears asking for a project name, as shown in Figure 3.6 (Windows), Figure 3.7 (Mac OS), and Figure 3.8 (Solaris). Enter a name

for your new project, and use the dialog box controls to navigate to a location on your hard disk where you want to save the project. Then click **Save** to save the new project information to disk.

TIP: We suggest naming your project with a `.mcp` file name extension. This makes your project easier to visually identify on your hard disk. In addition, the IDE uses this extension to quickly identify the project file when the project is moved across platforms.

Figure 3.7 Naming a new project (Mac OS)



WARNING! If you selected **Create Folder** when choosing stationery, and you already have a project with the same name in the same location on the hard disk, you will get an error message. Be sure to use a unique name for your new project.

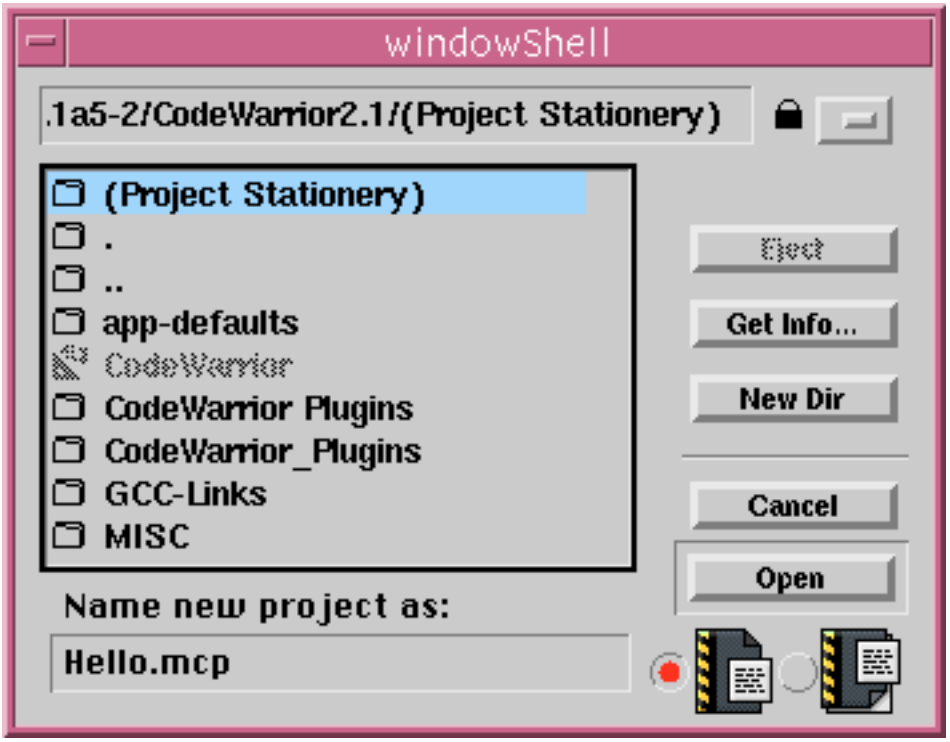
When you click **Save**, the CodeWarrior IDE automatically sets up the project, including:

- Creating a project folder with the same name as your project (if you selected the **Create Folder** checkbox as discussed in "Choosing the New Project Stationery File" on page 51),

without the file name extension. The new folder contains your new project file for the stationery you chose.

- Setting Preferences and Target Settings to be the same as the settings stored in the chosen stationery.
- Opening the project window. The new project contains libraries, source code placeholders, and resource file placeholders. If you chose to create an empty project, there will not be any files or libraries in your new project window.

Figure 3.8 Naming a new project (Solaris)



Modifying Your New Project

Most new projects created from stationery contain source files that are basically empty placeholders. You probably want to delete these files and replace them with source files of your own. See the section "Managing Files in a Project" on page 68 for more information about manipulating files in a project

You may also want to add additional libraries to your project file. To learn about which libraries to include, refer to the targeting manual of interest to you. Table 1.2 on page 25 lists all the targeting manuals for CodeWarrior.

Building Your New Project

After you have created your project and added your own files to it, you will want to build it to produce your target application, library, or whatever you are creating. To learn how to build a project, refer to "Compiling and Linking Overview" on page 317.

Working with Project Stationery

This section discusses how to create and use project stationery. It includes an explanation of project stationery that you can use to customize the project creation process described in "Creating a New Project" on page 50.

CodeWarrior projects can be configured to have multiple targets, and may also contain subprojects. To learn more about these topics, refer to "Working with Complex Projects" on page 85 after reading the material in this section.

The topics in this section include:

- About Project Stationery
- About the Project Stationery Folder
- Creating Your Own Project Stationery

About Project Stationery

A project stationery file is typically a minimal, pre-configured "starter" project file. Think of it as a template, or blank slate, that is used to quickly create a new project. When you create a new project or open a project stationery file, the CodeWarrior IDE creates a new project and, optionally, a new folder for the project. It then copies all the files related to the stationery project to the new folder.

A stationery project includes:

- All option settings for the project
- All files included in the stationery project (libraries, source code files, and resource files)
- (Mac OS) All segmentation and grouping information, including segment loader settings (68K projects only) and names.

When you use a stationery file to create your project, all the necessary files can be put into a new folder with the same name as your project. After creating your new project from stationery, you can open it and begin writing code in the CodeWarrior IDE.

About the Project Stationery Folder

CodeWarrior provides project stationery for many different kinds of projects. Project stationery for common types of projects are located in folder nested within the project stationery folder.

The following files can be included in the project stationery folder and are recognized by CodeWarrior as stationery projects.

- Normally-saved projects
- Project stationery files
- (Mac OS) Aliases to projects

Windows The `Stationery` folder, inside the `CodeWarrior` folder on your hard disk, stores project stationery.

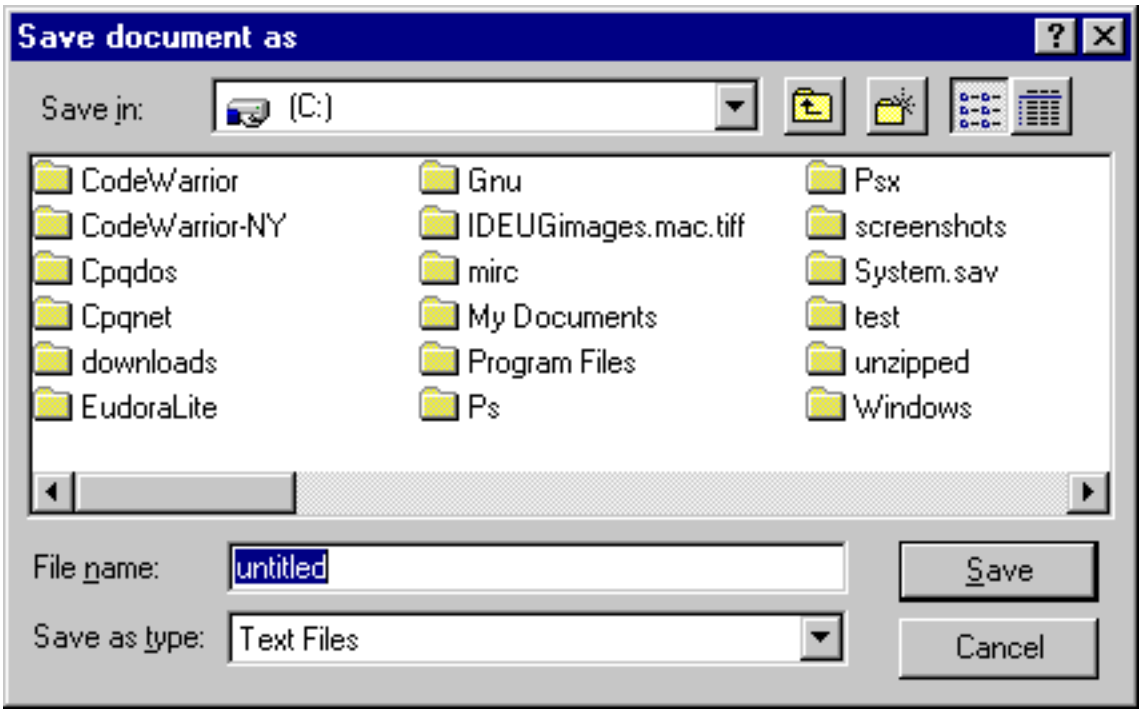
Mac OS The `(Project Stationery)` folder, inside the `Metrowerks CodeWarrior` folder on your hard disk, stores project stationery.

Creating Your Own Project Stationery

You can create a unique stationery or "template" project file that includes the files and options you want to have for a starter project. This stationery project can be reused whenever you are creating a new project, so that you always start from your own customized settings.

In essence, any CodeWarrior project you have can become a stationery project. To qualify as a stationery project it must: (1) be located in the project stationery folder, and (2) have its associated source files stored with it. When your project stationery is chosen in the New Project dialog, CodeWarrior duplicates the project and source files associated with the stationery project using the new project name.

Figure 3.9 Save A Copy As dialog box (Windows)

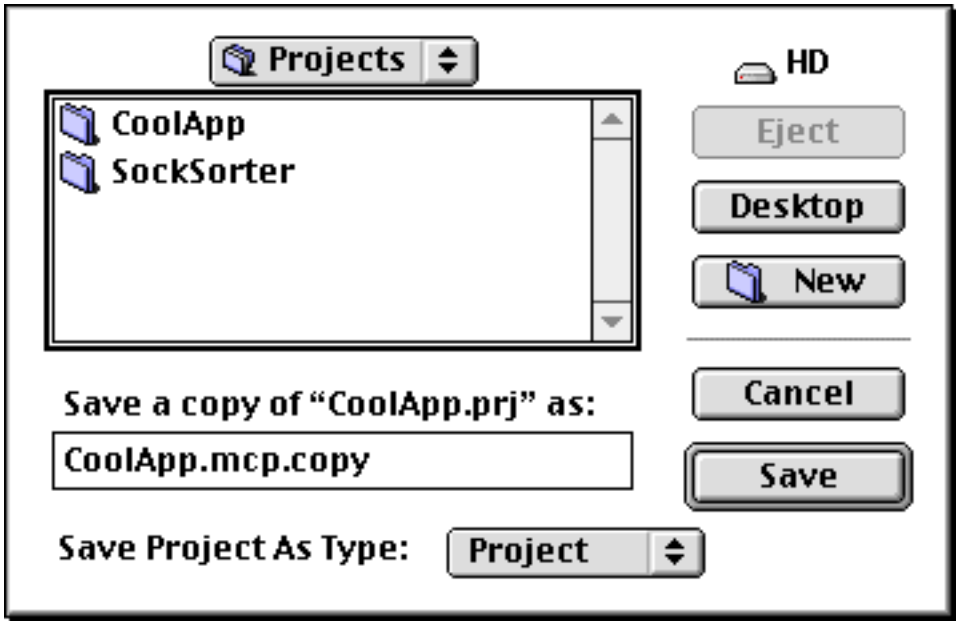


For beginners: Before creating your own project stationery, you should be familiar with the project stationery supplied with CodeWarrior. This will allow you to understand the importance of each of the files included with the project stationery.

To create your own custom stationery from scratch, start by creating a new project from an already available set of stationery, or just create an empty project. Save the new project to the project stationery

folder. Then, modify the stationery project's settings to best suit your requirements. Be sure to add and remove files as necessary to create the exact base project you want. Also, make sure a copy of all its source files are present in the new stationery project's folder, otherwise, the source files will not be copied to any new projects created with that project stationery. Make sure to save all your changes.

For beginners: **Figure 3.10Save A Copy As dialog box (Mac OS)**



By saving your stationery in one of the subdirectories, you will have your stationery available the next time you use the **New Project** command.

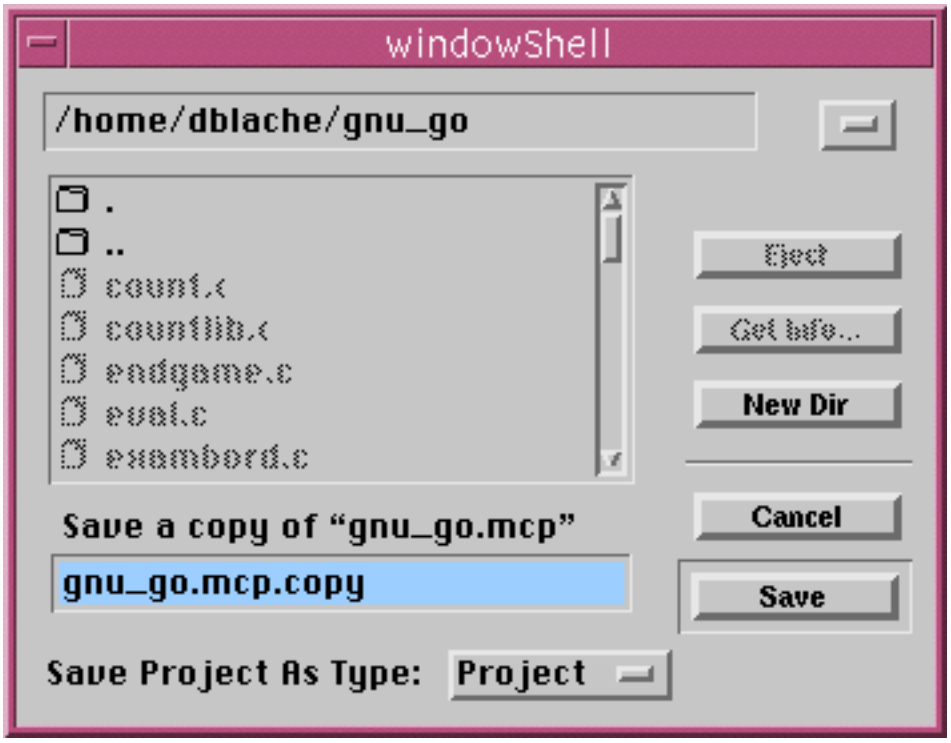
The next time you want to create a new project, choose your custom project stationery. All of its project and source files are copied to the new project's location, ready for modification.

To learn how to configure Preferences settings, refer to "Choosing Preferences" on page 238. You can also customize Target Settings for your custom stationery in a similar way. To learn more about how to do this, refer to "Choosing Target Settings" on page 293.

For information about adding or changing files in the project, see "Managing Files in a Project" on page 68.

See "Backing up files" on page 111 for more information about saving a copy of the project under a different name.

Figure 3.11 Save A Copy As dialog box (Solaris)



The reason you want to save the stationery project before doing a lot of work with it is because you will have a "starter" or "template" project file on your hard disk. You can make copies of the "starter" project to get new projects quickly started, with all your favorite settings, by using the **New Project** command.

New projects started with stationery will have all the settings you initially configured for your stationery project.

If at any time you decide that you want to use different project settings for new projects, you just create a new stationery project. Just make sure that you have a project window open, configure your op-

tions, and save your new stationery project in the appropriate stationery folder.

Opening an Existing Project

There are several ways to open a project file from within the CodeWarrior IDE. This section tells you how to open your projects so you can work on them.

Note that you may have many different projects open at a time, not just one project. See "Choosing a Default Project" on page 67 for more information on default projects.

The current project window you have open can be made the active window using the Window Menu. To switch to one of these opened projects, just choose a project from this menu.

The topics in this section are:

- Using the Open Command
- Using the Open Recent Command
- Using the Project Window to Open Subprojects
- Opening Project Files Created on Other Host Platforms
- Opening Project Files from Earlier IDE Versions

Using the Open Command

To open a Project File, choose the **Open** command from the File Menu. The IDE displays an Open file dialog box, as shown in Figure 4.1 on page 103 (Windows), Figure 4.2 on page 104 (Mac OS), and Figure 4.3 on page 104 (Solaris).

Windows If not already set, use the **Files of Type** pop-up menu to select **Project Files**. The list of files changes to show only project files that are eligible for you to open.

Mac OS The dialog displays a list of the available project and text files in the current folder.

Select the project file you would like to open, then click the **Open** button. The CodeWarrior IDE then opens the project and displays it in a Project window.

If the project was created with an older version of CodeWarrior, you will be prompted to update the older project. If you decide to update, a backup of the project is created before the updating takes place. See "Opening Project Files from Earlier IDE Versions" on page 63 for more information.

To learn more about working with CodeWarrior project files, see "Working with Projects Overview" on page 41.

You can have more than one project open at a time in the CodeWarrior IDE.

Using the Open Recent Command

The CodeWarrior IDE maintains some of the projects and files you have opened recently in a list under the File Menu. As a convenience, you may use the **Open Recent** menu command to reopen one of these projects.

To learn about setting the number of files that the IDE remembers in this menu, see "IDE Extras Panel" on page 242.

Using the Project Window to Open Subprojects

If your project contains subprojects, and you want to open one of those subprojects, you can double-click on the project file icon in the Project window to open it.

To learn more about subprojects, and how to add them to your Project window, refer to "Working with Complex Projects" on page 85.

Opening Project Files Created on Other Host Platforms

Project files are cross-platform compatible. For example, a project created on a Mac OS computer may be opened and used on a Windows computer.

To use a project created on another host platform, copy only its project file, not its associated `Data` folder, from the other host platform to your computer. After copying the project, open it in the IDE and recompile its files. Although a project's format is cross-platform compatible, its compiled object code isn't.

NOTE: Before copying a project make sure the project has a "`.mcp`" file name extension (without the quotes). The CodeWarrior IDE uses this file name extension to recognize project files. If the three letter extension is not present, the IDE will be unable to identify the project file.

TIP: When creating any project, always add the "`.mcp`" extension to the project file's name. This makes it easy to move project files to a different platform.

See also "Options Pop-Up Menu" on page 134 for information on editing source code files created on other platforms, and "Host Flags" on page 302 for information on setting up access paths for a target's host platform.

Opening Project Files from Earlier IDE Versions

The CodeWarrior IDE version 3.0 and later cannot use a project file from any version of CodeWarrior earlier than 1.7. You must recreate the project file from scratch using the new IDE, or convert the project file to work with the new IDE. This section discusses how you can convert projects.

Converting a single 1.7 project

The 3.2 CodeWarrior IDE has the ability to convert project files to the new format from the format used by 1.7 versions of the IDE.

Windows To convert a single 1.7 project, open it from the IDE. The IDE will prompt you to convert the project to the new format. The name of the converted project file will be based on the name of the original file.

Mac OS The IDE uses a conversion utility called Project Converter. The name of the converted project file will be based on the name of the original file. Project Converter is located in the `Other Metrowerks Tools` subfolder of the `Metrowerks` folder on the hard drive where CodeWarrior is installed. You can convert a single project in either of two ways:

- You can simply open the 1.7 project from the IDE. The 3.0 IDE and later versions will automatically run Project Converter to convert the project to the new format.
- Or, you can drag and drop the 1.7.x project file onto Project Converter itself.

NOTE: 1.7.x projects must be converted to the new format if you want to work with them in the 3.2 IDE, and the conversion is permanent. The 3.2 IDE cannot save a project in 1.7.x format.

Converting multiple 1.7 projects (Mac OS)

Project Converter has the ability to convert multiple 1.7 project files into a series of separate projects under the new format, or into one multi-target project file under the new format.

To take advantage of this, drag and drop a group of 1.7 project files onto Project Converter. You'll be asked if you want to keep them as separate projects, or merge them into a multi-target IDE project under the new format. Choose the option you wish.

Project Converter will prompt you to select a folder and name for the project or projects.

If you encounter problems, contact `support@metrowerks.com`.

You should read the Project Converter release notes file before converting your projects. You'll find this file in the `Metrowerks:CW-Pro Release Notes:IDE Notes` folder after CodeWarrior is installed. The release note will give you information on any late-breaking topics.

Opening project files from versions prior to 1.7 (Mac OS)

Project Converter only converts project files from the 1.7 IDE releases with CodeWarrior 10 and CodeWarrior 11. If you have an earlier project, you must convert it to CW10 or CW11 format before using Project Converter.

To update a project to the 1.7 format, first make certain that the 3.2 IDE is not running before you launch the 1.7 IDE. Then launch the 1.7 IDE and open the project.

The 1.7 IDE will update your project. You can then run the updated project through Project Converter to update that project to the new format.

If you don't have version 1.7 of the IDE, you can find it on the CWPro Tools CD. Look for the "CW11 IDE and Prefs.sit" file in the `Other Metrowerks Tools` folder.

Saving a Project

The CodeWarrior IDE automatically updates and saves your project when you perform certain actions. This section discusses these actions that cause the project file to get saved.

Your settings get saved when you:

- Close the project
- Change Preferences or Target Settings for the project
- Add or delete files for the project
- Compile any file in the project
- Edit Groups in the project

- Remove Object Code from the project
- Quit the CodeWarrior IDE

You never have to manually save your project unless you want to create a copy of it, since the project is automatically saved each time it is closed, and also when other common actions are performed.

Items Saved with Your Project

When the CodeWarrior IDE automatically saves your project, it saves the following information:

- The names of the files added to your project and their locations
- All configuration options
- Dependency information (the touch state and interface file lists)
- Browser information
- The object code of any compiled source code files

Saving a Copy of Your Project

If you want to save a backup copy of a project file before you make some changes to the original, use the **Save A Copy As** command in the File Menu. The CodeWarrior IDE creates a copy of the project file under a new name that you specify, but leaves the original project file unchanged and does not change the currently-open project to use the new file name.

WARNING! Do not attempt to make a copy of an open project from the desktop. Always close the project before copying the project file to prevent it from being corrupted.

Closing a Project

After you have been working with your project, you may want to close it to work on another project, or to quit the CodeWarrior IDE application to work on something else.

To close a project, ensure the Project window is frontmost, then choose **Close** from the File Menu.

You don't have to close your project before quitting the CodeWarrior IDE application, since your project settings are automatically saved. To learn more details about saved projects, refer to "Saving a Project" on page 65.

The CodeWarrior IDE will allow you to have more than one project open at a time, so you don't have to close each project you are finished with before switching to another project. Just open your new project and begin working with it.

TIP: Having multiple projects open at a time consumes more memory on your computer, and also causes project opening times to lengthen slightly.

Choosing a Default Project

Since the CodeWarrior IDE permits multiple open projects, it is sometimes ambiguous as to what project is used when you perform a Make, Run, or other operation on your project. If the active window is a project window, that project will be used for any builds that are started.

However, source code files can be in more than one open project. You can specify a default project for builds by using the **Set Default Project** command in the Project Menu. In any ambiguous case when a source code file belongs to one or more open projects, the CodeWarrior IDE will operate on the default project you have chosen using the Set Default Project menu command.

The first project you open becomes the default project. If you close the default project, the default project will then be set to the Project window that is closest to the top.

Managing Files in a Project

This section discusses adding, moving, naming, organizing, viewing, marking for compilation, and removing files from your project. The topics in this section are:

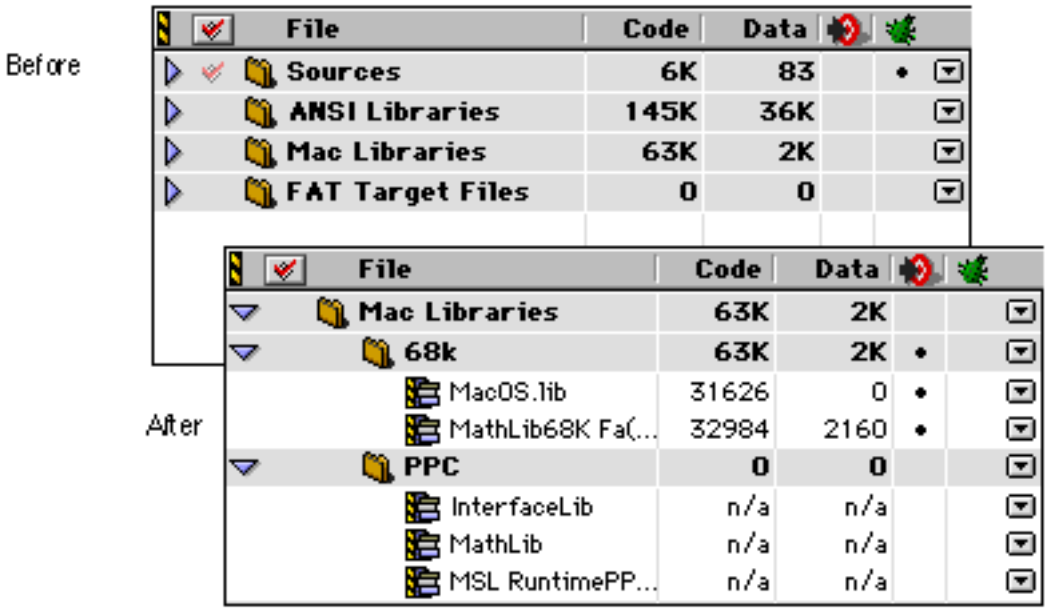
- Expanding and Collapsing Groups
- Selecting Files and Groups
- Adding Files
- Moving Files and Groups
- Creating Groups
- Removing Files and Groups
- Renaming Groups
- Touching and Untouching Files

Expanding and Collapsing Groups

Groups display files in collapsible lists. Click the tree control or disclosure triangle depending upon your platform to display or hide the contents of the group.

To expand a group and view its files, click the control in the top left edge of the desired group. To close a group and view only its name, click the control again.

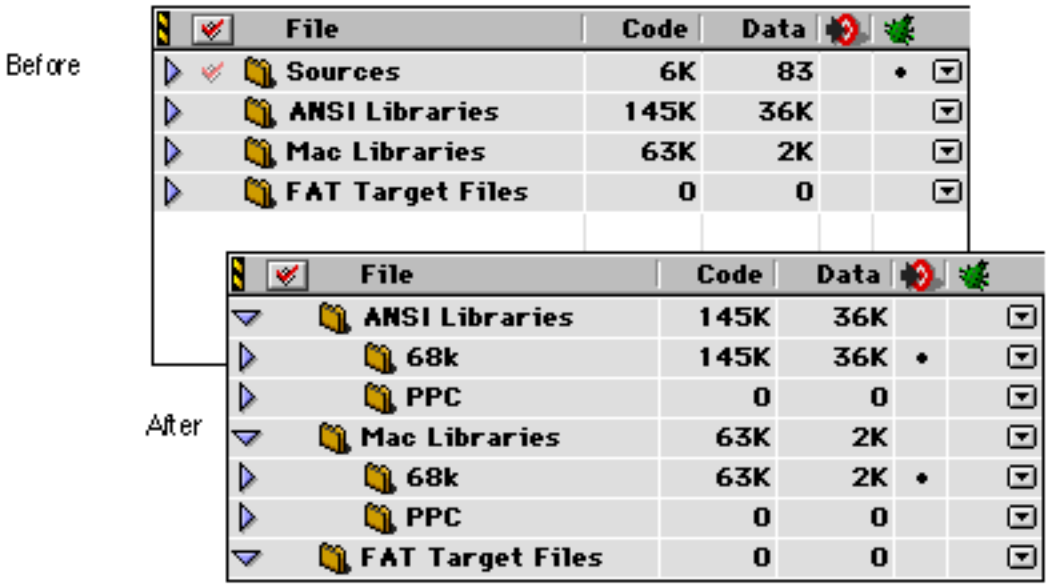
Figure 3.12 Expanding groups and subgroups



Use Alt/Option click to expand a group and all its subgroups in the project window as shown in Figure 3.12. It does not expand groups that are at the same level as the chosen group. Alt/Option click again to collapse the group.

Use Control/Command click to expand all groups at the same level in the project window as shown in Figure 3.13. Control/Command click again to collapse the sibling groups.

Figure 3.13 Expand all sibling groups



Selecting Files and Groups

From the project window you can select one or several files and groups to open, compile, check syntax, remove from the project, or move to a different group.

When a group is selected, all of the files within the group are selected, regardless of whether or not one of its files are included in the selection.

Selection by mouse-clicking

To select a single file or group in the project window, click its name.

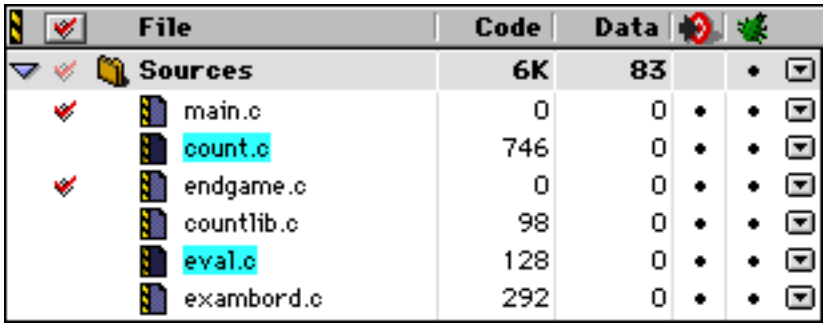
To select a consecutive list of files or groups, select the first file or group in the list by clicking its name, then Shift-click the last file or group to be selected.

Everything between and including the first file or group you clicked on and the last file or group you clicked on will be selected, whether they're groups or file names.

Another way to select a consecutive list of items is to drag-select them, like you would select items on the desktop.

Use Control/Command click to select or deselect any non-consecutive file or group in the project window as shown in Figure 3.14.

Figure 3.14 Non-consecutive file selection



Selection by keyboard

With the project window frontmost, you can type the first few characters of the name of the item you want to select. As you type, the CodeWarrior IDE selects the file in the project window as soon as the characters identify the file closest to your entry.

Use the Backspace/Delete key if you make a typo.

Use the Enter/Return key to open a file.

NOTE: Only files in currently-expanded groups in the Project window can be selected this way. Files in collapsed groups will not be matched with your keystrokes.

Adding Files

This section tells how to add files to your IDE project.

Here are the topics you will learn about in this section:

- Where files appear—where files go when they are added to your project
- Using the Add Files command—add one or more files
- Using drag and drop—add one or more files
- Using the Add Window command—add one file

When adding a file to a project, the Access Paths to the file or files automatically get set in the project. The Message Window informs you whenever a new access path is added.

NOTE: If you create a project using the **Empty Project** option, see "Linker" on page 297 and "File Mappings" on page 307. These sections describe how to configure the IDE so that you can add files to your project.

Normally, the IDE forces each file in a project to have a unique name. To remove this restriction, turn on the **Save Project Entries Using Relative Paths** option in the Target Settings panel.

For more information about Access Paths see "Access Paths" on page 298.

For more information about the Message Window, see "Guided Tour of the Message Window" on page 337.

Where files appear

Files are always added after the currently selected item in the Project window, or at the bottom of the Project window if there is nothing selected. To put a new file or files in a particular location, always select the file or group above that location before performing the **Add Files** or **Add Window** command.

If a group is selected, regardless of whether or not the group is expanded or collapsed, the files to be added are placed at the end of the selected group. To learn about how to select a file or group of files, see "Selecting Files and Groups" on page 70.

NOTE: If a group or a file within a group is not selected prior to adding files, a new group is created and appended to your project. The files added are placed in this new group.

Of course, you can always move a file or group of files to a new location after adding to the project. To see how to move files and groups around in the project window, see "Moving Files and Groups" on page 79.

Using the Add Files command

The **Add Files** command on the Project Menu opens a dialog box you can use to add files to your project from many locations. Use this command to add source code files, libraries, resource files.

This command opens a dialog box to add files to your project from a directory. In order for files to appear in this Add Files dialog box, the files must have a recognized extension (such as `.c` or `.p`) at the end of the file name. To examine and configure possible extensions for file names, refer to "Setting a File Extension" on page 319.

Add Files (Windows)

The Files of Type pop-up is automatically set to view all file types.

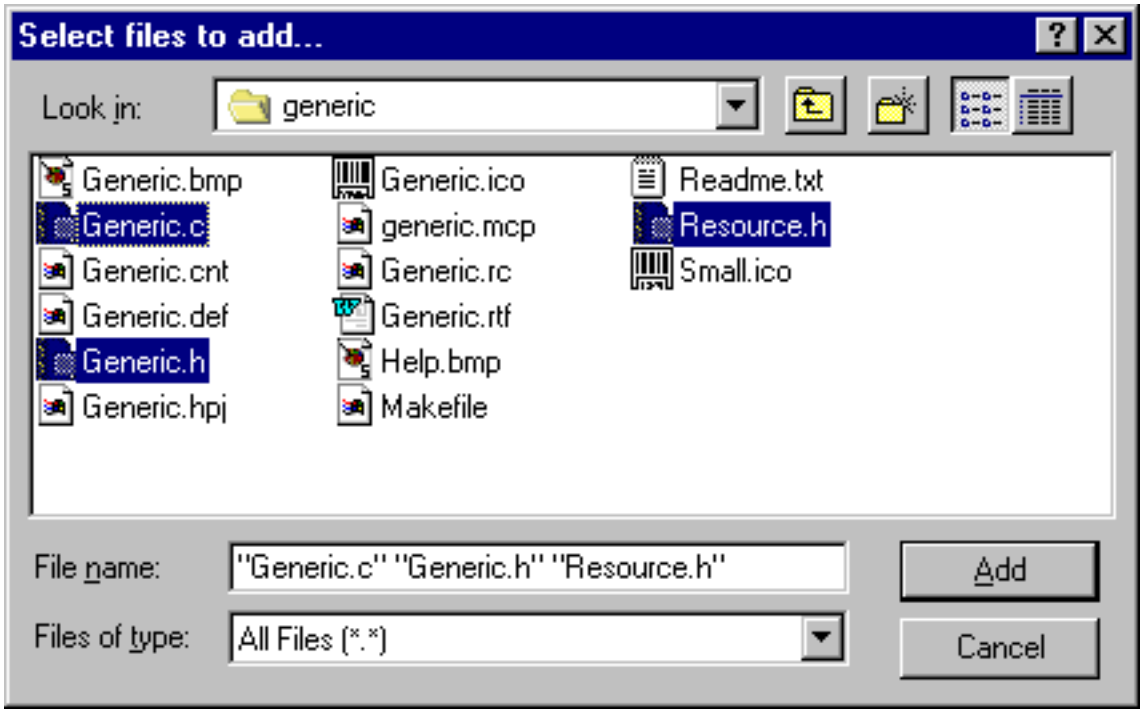
Select the file in the Files List that you wish to add to your project, then click the **Add** button.

To select multiple files, as shown in Figure 3.15, press the Control key when clicking on a file name in the dialog box.

To select a contiguous group of files, click on the first file name in the group, then press the Shift key and click on the last file in the group.

When all the files you want to add are selected, click the **Add** button.

Figure 3.15 Adding files to a project (Windows)



Add Files (Mac OS)

This dialog box is split into two lists, as shown in Figure 3.16.

Use the pop-up menu at the top of the Files List to navigate through folders and drives to find files to add. The Select Files To Add List shows the files you have added to your project.

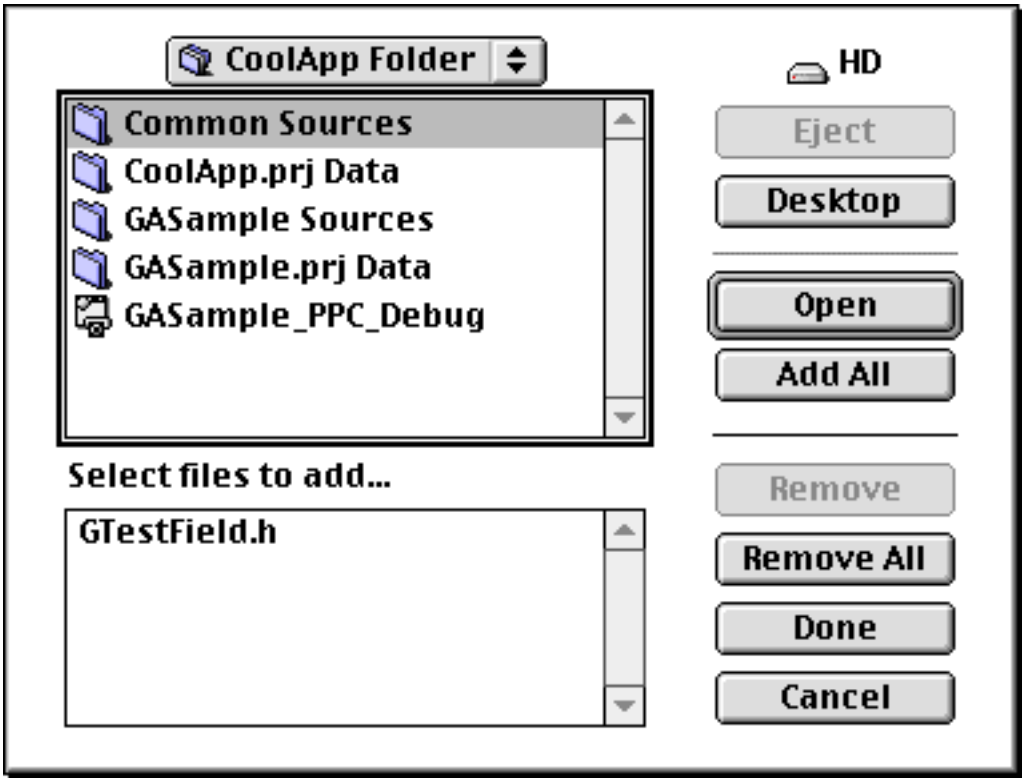
Select the file in the Files List that you wish to add to your project, then click the **Add** button. The selected file is added to the Select Files To Add list. Alternatively, you can double-click the selected file to move it to the Select Files To Add list.

To add all the files displayed in the file list to the Select Files To Add list (all available files in the folder), click the **Add All** button. This does not add files stored in any subfolders shown in the file list.

To remove a file from the Select Files To Add list, select the file to be removed and click the **Remove** button.

To clear the Select Files To Add list, click the **Remove All** button. The **Cancel** button closes this dialog box without adding any files to the Project window.

Figure 3.16 Adding files to a project (Mac OS)



When you've finished selecting all the files you want to add from the currently selected directory, click **Done**. The files shown in the Select Files To Add list are added to your project. You can add large numbers of files this way, but there may be a delay while the CodeWarrior IDE locates the files and adds them to the project.

NOTE: (Mac OS) Beginning with Mac OS 8.5, the dialog box that you use to add files to a project will no longer contain a secondary list as shown in Figure 3.16. Instead, you will see a dialog box with a single list. To select a file to add to your project, locate that file in the list, highlight it, and click the **Open** button. To select multiple

files to add to your project, click on the disclosure triangles in the list so that the files you wish to add are visible. Shift-click each file that you want to add to your project, then click the **Open** button. All of the highlighted files will be added to your project.

Note that if your project contains multiple targets, you will be prompted to choose the targets that your files should be added to.

Add Files (Solaris)

The file path text box at the top of the dialog box displays the path to the current directory being listed.

Select the file that you wish to add to your project and click the **Add** button. The selected file is moved from the file list to the Select Files To Add list. Repeat this procedure for each additional file that you wish to add to your project.

When all the files you want to add have been chosen, click **Done**. The files in the Select Files To Add list are then added to your project.

Using drag and drop

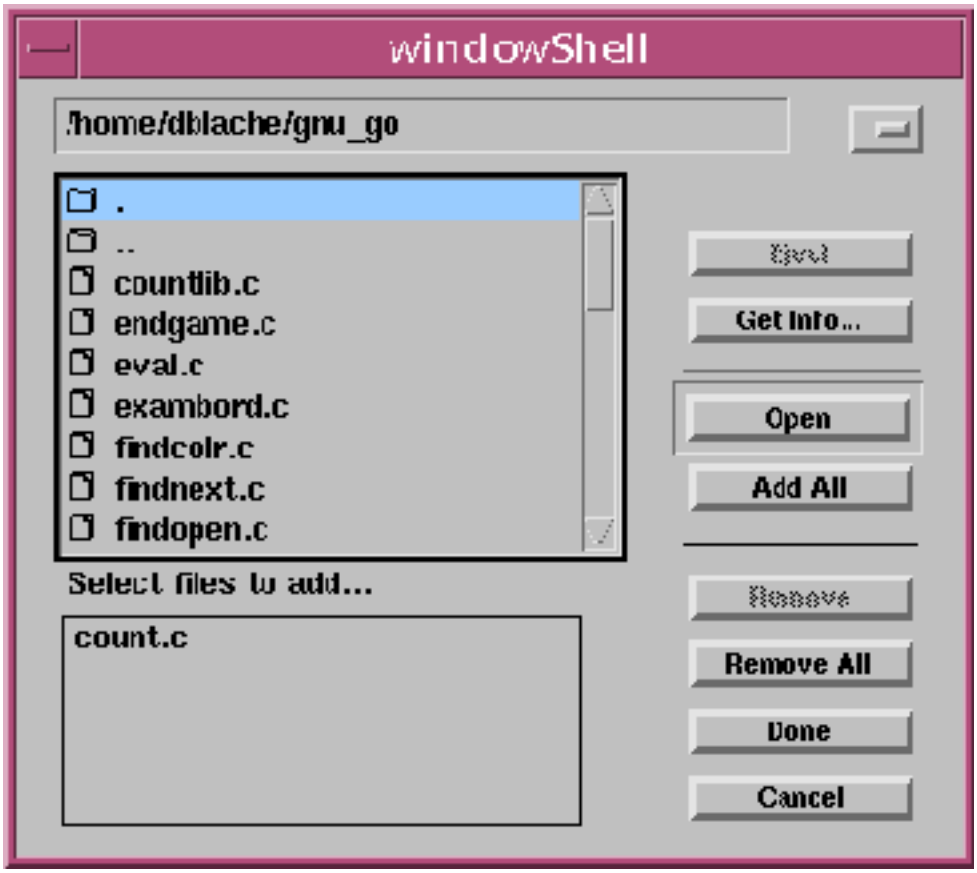
When you drag and drop files or folders onto an IDE project window, they will be added to the project.

To add files to your project using this method, first select the files or folders you want to add to the project.

You can select files in many places, including the desktop, or the multi-file search list in the CodeWarrior IDE's Find dialog box.

To complete the add operation, drag your selection onto the project window. Note that if your project contains multiple targets, you will be prompted to choose the targets that your files should be added to.

Figure 3.17 Adding files to a project (Solaris)



When dragging the selected files onto the project window, the CodeWarrior IDE verifies that the files can be added to the project. When dragging a folder, the CodeWarrior IDE checks to make sure that the folder, or one of its subfolders, contains at least one source code file, library, or resource file, and that file is not already in the project.

If the selection does not contain at least one file recognized by the CodeWarrior IDE, the IDE won't accept the drag.

NOTE: See "Linker" on page 297 and "File Mappings" on page 307 for more information on configuring the IDE to recognize files.

Use the focus bar (an underline) that appears in the Project window to select the location where files will be inserted into the project list.

Releasing the mouse button (dropping the files) adds the dragged items to the project, inserting them below the position specified by the focus bar. If there are multiple targets in the project file, a dialog box (Figure 3.18) opens that enables you to choose targets that receive the files.

To create a new group and add files to it, drop the files when the cursor is over the blank space after the last group.

The CodeWarrior IDE does not allow the dragging of entire volumes (such as your hard disk) onto the project window.

The CodeWarrior IDE allows dragging and dropping outside of the project window. You can also drag files to another application to open them in that application.

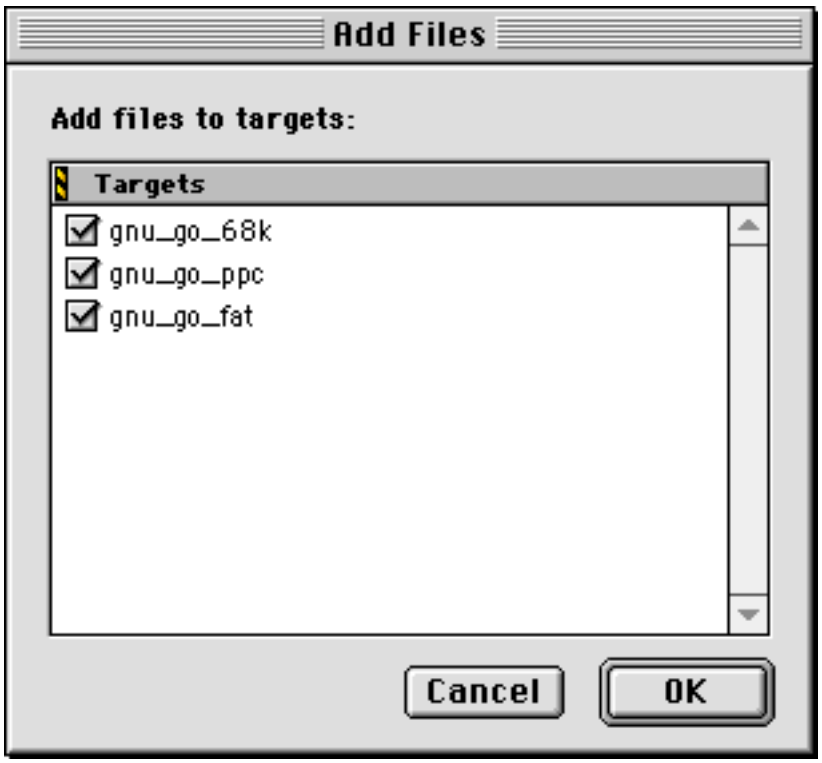
Although the CodeWarrior IDE supports dragging and dropping files into the project window on all platforms, some do not allow you to remove files by dragging them out of the project window. To learn how to remove files from the project window, refer to "Removing Files and Groups" on page 81.

Using the Add Window command

The Add Window command adds the file associated with the active editor window to the project. You typically use this command when you've created a new file and decided that you would like to add it to the active project.

To use the Add Window command, select a location in the project window. Then, open a source code or text file, make sure its window is active, and select the **Add Window** command from the Project Menu. If the window is untitled, the Save As dialog box appears, prompting you to select a location and name for the file. After you save the file, the file is added to the open project.

Figure 3.18 Add files to targets dialog.



Note that if your project contains multiple targets, you will be prompted to choose the targets that your files should be added to (Figure 3.18).

NOTE: The Add Window command is enabled when the active window is a text file, the file is not yet in the project, and it has a recognized file name extension (to learn about configuring permissible file name extensions, refer "File Mappings" on page 307). The Add Window command is dimmed otherwise.

Moving Files and Groups

To move one or more files or groups within a project's File view, or to arrange build targets in a project's Targets view, select the files or groups to be moved. Selecting a group includes all of the files in that

group, regardless of whether or not those files are visually selected in the project window. If you need help selecting files and groups, see "Selecting Files and Groups" on page 70.

Next, drag the selected files or groups to their new location in the project window.

A focus bar (an underline) indicates where the selected files will be moved when the mouse button is released.

Whether you are moving files or groups depends on your selection. For example, if your selection consists of files then the focus bar is shown on each line, under both groups and files.

If your selection includes at least one group, then the underline is shown only under other groups as you move the mouse in the project window, allowing you to rearrange groups.

Finally, release the mouse button when the focus bar is positioned after the desired file or group position.

When a file has the focus bar underneath itself, and the mouse is released, the selected files are placed in the same group, following this file. When a group has the focus bar underneath, the selected files are placed at the end of the chosen group.

TIP: The focus bar has a small arrow at the left end that indicates the level of insertion. If the arrow is to the left of a group icon, the insertion will be at the same level as the target group. If the arrow appears to the right of the icon, the files are inserted into the target group.

Creating Groups

With the Project window frontmost, and the File view visible, choose the **Create New Group** command from the Project menu. Enter a name for the new group in the Create Group dialog box, then press **OK**. For information on how to change this name, see "Renaming Groups" on page 82.

Dragging items past the last group adds them to the top level of the window, instead of creating a new group.

Mac OS The new group may use "Segment" in place of "Group" if the project is targeted for the 68K microprocessor.

Removing Files and Groups

There are two methods used to remove files from a Project window. You can use menu commands or drag and drop on supported platforms to remove files from a project window.

Using menu commands

You can remove files from either the File view or the Link Order view of the project window. When removing files from the project window, you need to be aware of a subtle issue. If you remove files from the Files view, *they are removed from the entire project, including all targets*. When removing files from the Link Order view (also called the Segments view or Overlays view for some targets), the files are removed from the current target.

To learn more about targets, refer to "What is a Build Target?" on page 86.

To remove one or more files or groups in the Files view, first select the files or groups to be removed. Note that selecting a group includes all of its files regardless of whether or not those files are visually selected in the project window.

To learn how to select files, refer to "Selecting Files and Groups" on page 70.

After selection in the Files view, choose the **Remove Selected Items** command from the Project Menu or press Alt/Option-Delete. All the selected files and groups are removed from the project.

WARNING! This command can't be undone. If you mistakenly remove a group, you must re-add its files using either the **Add Window** or **Add Files** commands under the Project Menu.

To remove a group from your project in the Files view without removing the files that are in the group, select the group to remove and press Shift-Ctrl/Command-Enter/Return. The CodeWarrior IDE moves the files in this group up to the previous group.

When you remove a group in the Links view, all of the files that are in that group will still be in the project. If there is more than one group in the project, the files are added to the group that is above the group you are deleting.

If there is only one group in your project, whether in the Files or Link Order view, you won't be able to delete the group, since the CodeWarrior IDE requires at least one group in a project.

Using drag and drop

The CodeWarrior IDE supports dragging and dropping files into the project window, although you cannot drag files out of the project window on some platforms.

Renaming Groups

To rename a group, select the group to be renamed by clicking on it then press the Enter/Return key. You may also use the arrow keys to navigate to the group, then press the Enter/Return key. A dialog box appears (Figure 3.19) containing a name text field you use to enter a new group name.

Figure 3.19 Changing a group name

Type the new name in the Enter Group Name field and click **OK**. The name of the group is changed in the project window.

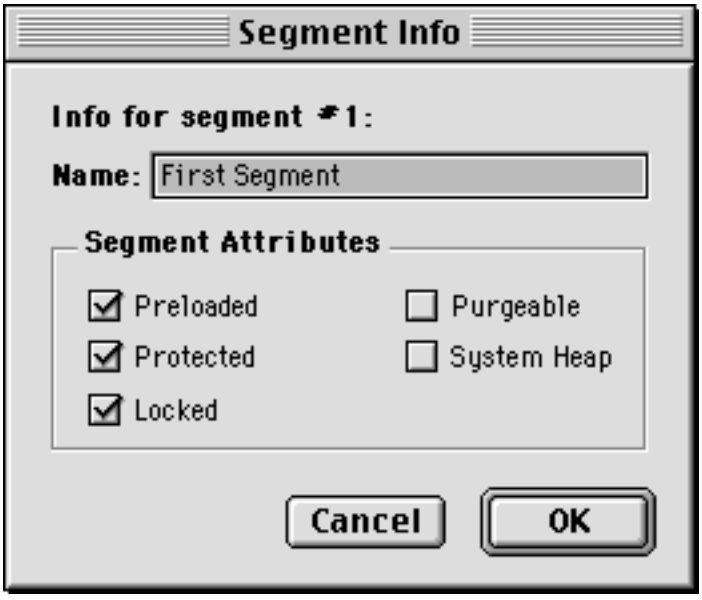
If you have selected more than one group, the same dialog box opens for the next group selected, enabling you to change its name as well.

Mac OS On Mac OS 68K targets, the dialog box in Figure 3.20 will appear. It contains a series of checkboxes that affect segment loader information. The Mac OS 68K linker gives this name to the code segment when the project is built. See the *Targeting Mac OS* manual for more information on using segments.

Touching and Untouching Files

Use the Touch column shown in Figure 3.2 on page 45 to mark files that need compilation. The CodeWarrior IDE doesn't always recognize file changes and may not automatically recompile all files in certain cases, which is why the Touch column features are useful.

Figure 3.20 Changing a segment name



There are three possible ways to make sure files get compiled. One way is to click in the Touch column beside the file name in the project window's File view. A check should appear in the Touch Column next to the file name.

Another way is to select the **Touch** command from the Interface pop-up menu. The Touch command may appear at the top of the Interface pop-up.

The last way to make sure that changed files get compiled is to click on the Touch Column icon at the top of the column to resynchronize the state of the files in the project depending on the dates they were last modified. This is useful if the files have been modified outside of the CodeWarrior IDE, perhaps by a third-party editor.

TIP: If the file hasn't been changed since it was last compiled, the first command in the Interface pop-up is **Touch**. When you choose Touch, the CodeWarrior IDE marks your file to be compiled the next time it makes your project. If the file has been changed since it was last compiled, the **Untouch** command is shown.

To unmark files so that they won't be compiled, click again in the Touch column left of the file name, or choose **Untouch** from the Interface pop-up.

Note that the "check" icon at the top of the Touch Column may be used to touch all the files in the entire project.

Synchronizing modification dates

To update the modification dates stored in your project file, click the icon above the Touch column. Alternatively, choose the **Synchronize Modification Dates** command in the Project Menu.

The CodeWarrior IDE updates the modification dates stored in the project file. It checks the modification date for each file in the project, and if the file has been modified since it was last compiled, the CodeWarrior IDE marks it for recompilation. This will resyn-

chronize the state of the files in the project depending on the dates they were last modified. This is useful if the files have been modified outside of the CodeWarrior IDE, perhaps by a third-party editor that doesn't notify the CodeWarrior IDE when it modifies a file.

NOTE: (Mac OS) Some third-party editors use AppleEvents to let the CodeWarrior IDE know when you modify a file. These editors include BBEdit from BareBones Software, Object Master, and Alpha. You don't need to use the Synchronize Modification Dates command if you use one of these external editors.

Working with Complex Projects

The CodeWarrior IDE provides flexible facilities for creating project files that use sophisticated build rules. This section discusses how to construct complex project files that may contain different kinds of build target code, or contain other projects. This facility allows you to create powerful build hierarchies for your entire software project.

For example, you may want to create complex projects so that one project file can contain build targets for both shipping and debugging versions of your software. By switching between shipping and debug build targets, the IDE generates different versions of the software during the development process. Each of these build targets can have their own settings. For example, the debugging build target could have optimizations disabled and debugging information enabled, and the shipping build target can have code optimizations enabled.

The topics in this section are:

- What is a Build Target?
- What is a Subproject?
- Strategy for Creating Complex Projects
- Creating a New Build Target
- Changing a Target Name

- Changing the Target Settings
- Setting the Current Build Target
- Creating Target Dependencies
- Assigning Files to Targets
- Creating Subprojects Within Projects

What is a Build Target?

A build target is a set of rules and settings that you configure to produce a an output file, such as an application or library.

The CodeWarrior IDE has the capability to build many different kinds of output files, or targets, from one project file, as shown in Figure 3.4 on page 50. For example, this is useful if you want to have a build of your code for debugging, and a separate build for your shipping code.

You can also define build targets that are common to multiple targets, so that one target gets built before trying to build another. This could be useful for sharing resource files between other targets. In this way, you can create a target that depends on some other target, forcing the latter target to build first.

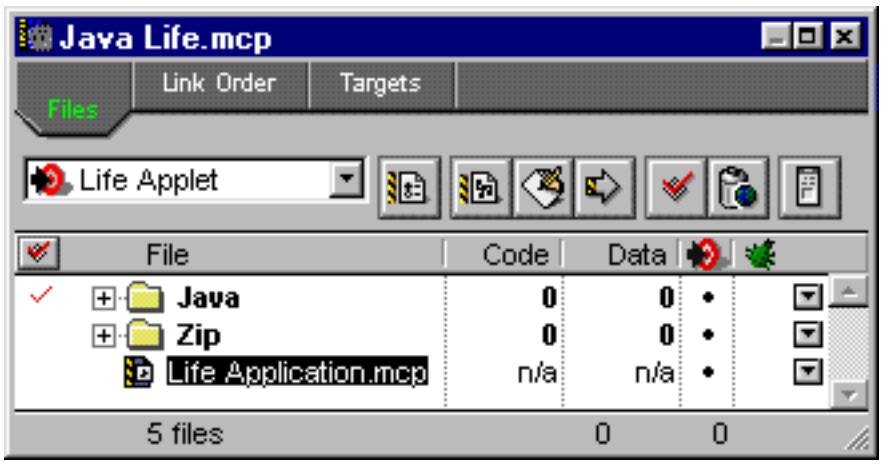
Each target in the project can have its own build settings. Each set of target settings is distinct.

To learn more about considerations for using targets, refer to "Targeting Documentation" on page 24 and "Strategy for Creating Complex Projects" on page 88.

What is a Subproject?

A subproject is a project file that is nested within a project, like the project named `Life Application.mcp` shown in Figure 3.21. Subprojects are useful if you have a project file that you want to keep separate from the main project file. This allows you to organize the build process into separate project files.

Figure 3.21 Subproject within a project



One case where this organization might be useful is for developing applications that use a plug-in architecture. Suppose your program uses many different plug-in modules, each sharing some common source code with other plug-ins. You might create one project file to build all the plug-ins by creating a separate target for each plug-in. In this scenario, the project file for the plug-ins is the subproject. Including the subproject in the project file of the main application causes the subproject to be built, before building the main project.

A project file may be assigned to any target in a project. To learn how to do this, refer to "Assigning Files to Targets" on page 92.

To learn how to add files to a project, refer to "Adding Files" on page 71.

You may select one or more build targets in a subproject, to be built when the containing target in the main project is built. When the target in the main project is built, the CodeWarrior IDE first builds any selected targets in any subprojects. You can optionally link the output of the main project with the output of the subproject's target by selecting the marker for the target in the Link column of the main project's Targets view. For information on the Link column, see "Targets View" on page 50.

A subproject's targets are not built automatically when a subproject is added to a parent project. Only the chosen targets within the subproject will be built.

Subprojects can be made target-specific. That is, if you add a subproject, you can choose which targets it belongs to. Other targets in the main project will not build the subproject unless the subproject file is added to the target you choose.

Strategy for Creating Complex Projects

The choice of whether to use multiple targets or subprojects within a project file depends on what works best for you. If you want access to all the source code in one project, then using multiple targets is a good choice. Subprojects are better when you prefer to keep separate stand-alone project files.

For example, if you need to build a number of plug-in libraries that accompany your application, create a project that builds the subprojects with a single Make command. Then, include this project file as a subproject in your main application project file. When your main application is built, the subproject's plug-ins will be built first.

There is a limit of 255 targets per project. Before you hit that limit, there's the consideration of memory and project load times. Projects with lots of targets will take up more disk space, take longer to load, and use more memory.

Once you get past ten or twenty targets, it's likely that you would benefit by moving some of them off to subprojects. Anything that is not built often and uses a distinct set of source files is a good candidate for moving to a subproject.

Creating a New Build Target

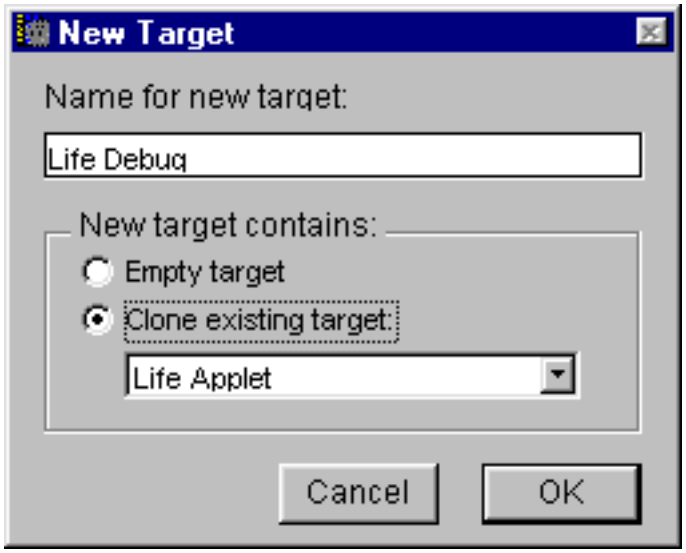
To create a new target in your project, use the **Create New Target** command in the Project Menu. This command appears if you have the Targets view selected in your project window, as shown in Figure 3.4 on page 50.

After you choose this menu command, you see the dialog box as shown in Figure 3.22. In this dialog box, you can choose the name of the new target using the Name For New Target editable text field.

Then, choose whether you want your new build target to be empty, or a clone of a previous target. If you choose **Empty Target**, you need to configure all the settings of the target as if a new project window were just opened. If you choose **Clone Existing Target**, the settings for the new target are the same as those of the target that you chose from the pop-up menu. You also get a copy of all the files that the original target contains.

After creating a target, you may want to associate the target with other targets, in order to create dependent build relationships. To learn how to do this, refer to "Creating Target Dependencies" on page 91.

Figure 3.22 New Target dialog box



To learn how to configure settings for a target, refer to "Choosing Target Settings" on page 293.

When creating a new target that depends on using an output file from another target, you will need to click in the Link column for the target that creates the output file.

Changing a Target Name

To change the name of a target in the Targets view of the project window, double-click the name of the target to open the settings dialog box for that target, as shown in Figure 9.4 on page 296. Select the Target Settings panel from the list of available panels. Change the name of the target using the Target Name editable text field.

You can also open the settings dialog box using the **Target Settings** command in the Edit menu. The actual name of the Target Settings command will include the name of the target.

To learn more about the Target Settings panel, refer to "Target Settings" on page 295.

Changing the Target Settings

Each build target in a project has its own settings. You modify these settings through the settings dialog box.

To open this dialog box for a particular target in your project, just double-click the name of the target in the Target view of the project window. Or, use the **Target Settings** menu command in the Edit menu. The actual name of the Target Settings command will include the name of the target.

The left side of this dialog contains a list of all settings panels appropriate for this target. Select a panel to see the options you can set.

To learn how to change these, refer to "Configuring Target Options Overview" on page 289. All of the generic settings panels are described in that section of this manual. Settings panels that are specific to a particular operating system are described in a corresponding Targeting manual, such as *Targeting Mac OS* or *Targeting Win32*.

Setting the Current Build Target

You can choose a different target within the current project to work with by using the **Set Current Target** command under the Project Menu. This command is be useful if you want to switch between multiple targets in a project, and do a build for each one.

You can also change the current target by using the Targets view of the Project window, as shown in Figure 3.4 on page 50. The current targets that will be built are denoted by the circle icon (an archery "target") with an arrow going into it. To change to a different target for building, click once on the name of the target you want to choose as the current target.

Creating Target Dependencies

You can configure a target to depend on other targets. These dependencies are useful when you want to ensure that certain targets build before others. For example, you could have a target that depends on information in a second target. In order to compile correctly, the second target should be built before the first target.

To specify that a target depends on a second target, first go to the Targets view of the Project window. Drag the second target below and indented to the right of the first target's entry. The IDE adds an italicized entry within the first target's group for the second target. This makes the first target dependent upon the presence of the second target. Now, when the IDE builds the first target, it ensures that the second target is built before attempting to build the first target.

To specify that the first target should be linked with object code from the second target, click the marker in the Link column of the second target's italicized entry within the first target's entry.

If you don't know how to create new targets, refer to "Creating a New Build Target" on page 88 to learn how.

Refer to "Setting the Current Build Target" on page 91 to learn how to set the current target before building if you don't already know how to do so.

To learn more about strategies for setting up complex projects using targets and subprojects, refer to "Strategy for Creating Complex Projects" on page 88.

Assigning Files to Targets

There are two ways to assign files to targets in a project: with the Target column in the File View and with the Project Inspector.

Assigning files to targets with the Target column

The Target column in the Project window's File view indicates whether a file is in the current target or not. The IDE displays this column if the project has more than one target. A file in the File view has a marker in the Target column if it is in the project's current target.

To assign a file or group to the active target, click in the item's Target column to place a marker. To remove a file or group from the active target, click in its Target column to remove its marker.

To assign or remove all the items in the File view to the current target, Alt/Option-click in the Target column to display or erase markers.

Assigning files to targets with the Project Inspector

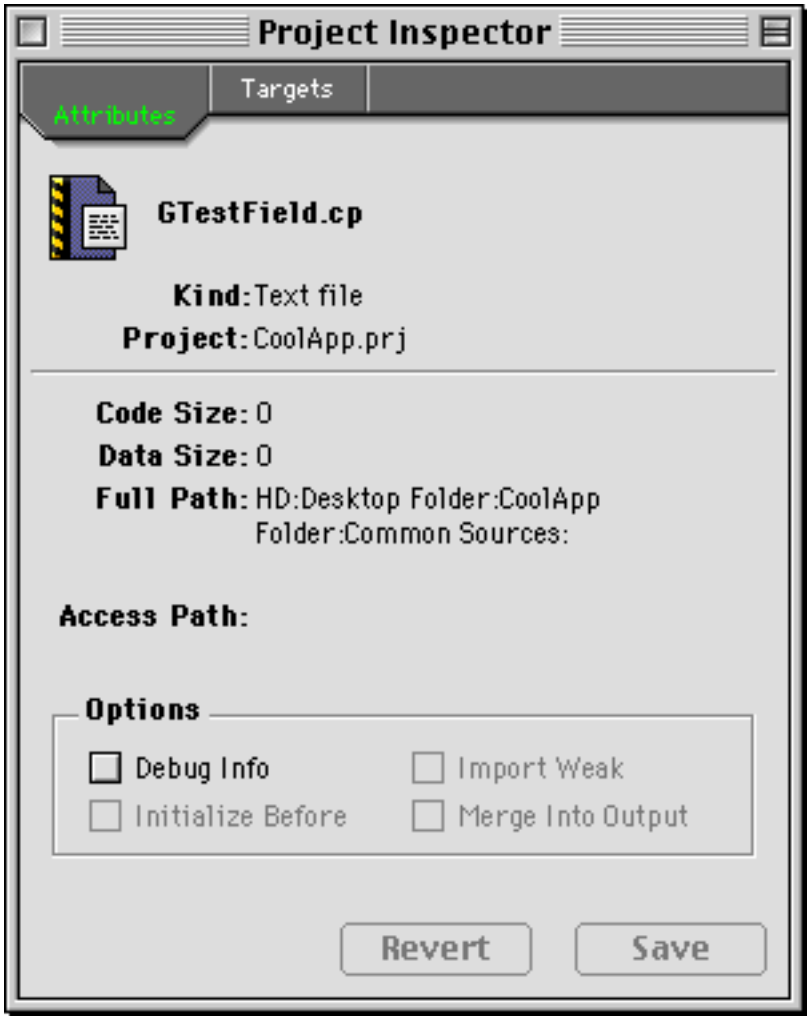
You can select the target that a file belongs to using the Project Inspector window. First, select a file in the project window. To learn how to do this, refer to "Selecting Files and Groups" on page 70. Then, choose the **Project Inspector** menu command from the Window Menu. A window appears, as shown in Figure 3.23.

Click the Targets tab to switch the view to that shown in Figure 3.24 on page 94. This window shows you the targets that your selected file belongs to. Click in the check boxes on the left side of the window to include or exclude the file from a given target.

You may close the window when you are finished by clicking the close box. If you make changes that you want to undo, click the **Re-**

vert button. If you want to apply your changes but keep the window open, click the **Save** button.

Figure 3.23 Project Inspector window for attributes



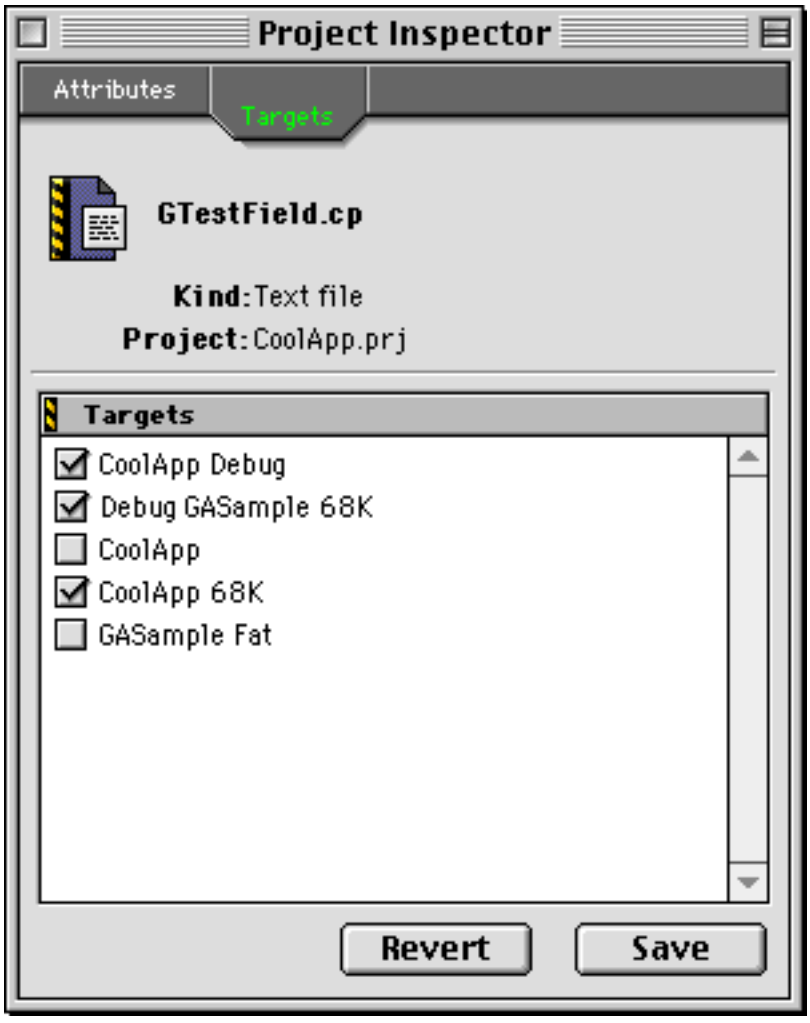
Creating Subprojects Within Projects

To create a subproject, just drag and drop a project file into an open project window. Note that if your main project file contains multiple targets, you will be prompted to choose the targets that the subproject should be added to. After you do this, the project window

will look similar to Figure 3.25 on page 95, with a project file added to the list of files in your project. You may also add the file using the methods discussed in "Adding Files" on page 71.

Adding a project to another project makes the subproject file part of the main project file. When you do a Make on the main project, the subproject will be built first.

Figure 3.24 Project Inspector window for targets



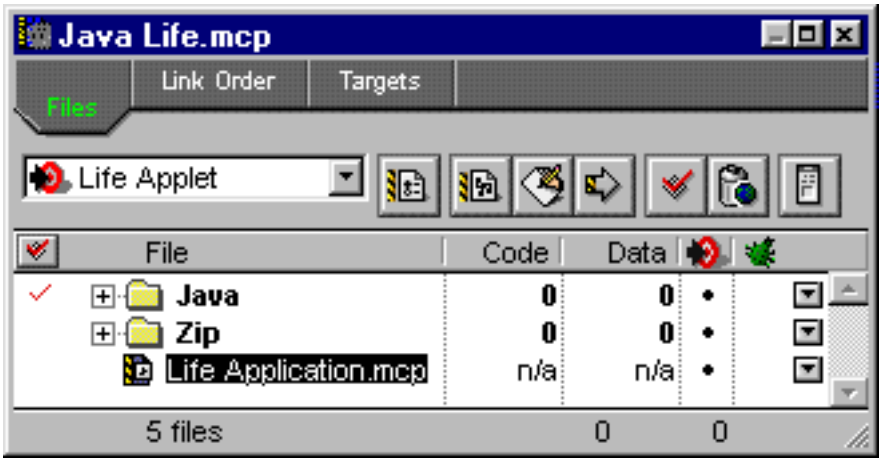
Once you add a subproject to a project, you can assign which targets the subproject is used in. To learn how to do this, refer to "Assigning Files to Targets" on page 92.

Examining Project Information

The CodeWarrior IDE allows you to review and configure information about your project and source code files, using the Project Inspector window, shown in Figure 3.23. To show this window, choose the **Project Inspector** command from the Window Menu.

There are two tabbed views in this window, Attributes and Targets.

Figure 3.25 Subproject within a project



To learn how to inspect and set the Targets for which a given file will be compiled, refer to "Assigning Files to Targets" on page 92, as that section of the manual discusses Targets in detail.

To learn more information about a given file in your project, you can review its Attributes settings. The Attributes include the file name, its path to a location on your hard disk, the name of the project it is included in, the code and data size (if it has been compiled), and whether or not Debug Info is being generated for the file when compiled.

You may close the window when you are finished by clicking the close box. If you make changes that you want to undo, click the **Revert** button. If you want to apply your changes but keep the window open, click the **Save** button.

To learn about configuring Debug Info for files, refer to "Controlling Debugging in a Project" on page 97.

Mac OS To learn more about setting Initialize Before, Merge Into Output, and Import Weak, refer to "Special Library Options (Mac OS)" on page 350.

Moving a Project

The CodeWarrior IDE stores all required information about your project in the project file. There are other files, usually stored in a folder having a name similar to your project file, that contain information about window positions, object code, debug info, browser data, and other settings which are not crucial to rebuilding the code of your project. These additional files are not needed by the CodeWarrior IDE to recreate your project.

WARNING! (Windows) Previous versions of the Windows-hosted CodeWarrior IDE used a folder called `Resource.frk` to store state information for the project file. `Resource.frk` is now obsolete and will no longer be generated or utilized by the CodeWarrior IDE.

To move your project on your hard disk, just copy the project file (ending in `.mcp` if it obeys the project file naming convention) to a new location on your disk. If you want all the information in the additional files to travel with the project file, you may also copy the folder containing those files. However, these files are not needed. The CodeWarrior IDE is able to reconstruct its state when a Bring Up To Date or Make operation is performed. Generally, you would only check the main project file into a revision control system, and not the other files.

If you have set absolute Access Paths, you may need to modify them when you move your project file. To learn how to do this, refer to "Access Paths" on page 298.

See also "Opening Project Files Created on Other Host Platforms" on page 63.

Controlling Debugging in a Project

Your program will probably not run correctly the first time you build it. In order to debug it, you need to enable debug information for your project and its files. This section tells you how to do this.

The topics in this section are:

- Activating Debugging for a Project
- Activating Debugging for a File

Activating Debugging for a Project

To enable debugging for a project, you need to set certain options in the Project Settings. If you select the **Enable Debugger** command from the Project Menu, all the necessary configuration is done for you automatically.

To learn how to do this manually, refer to "Choosing Target Settings" on page 293.

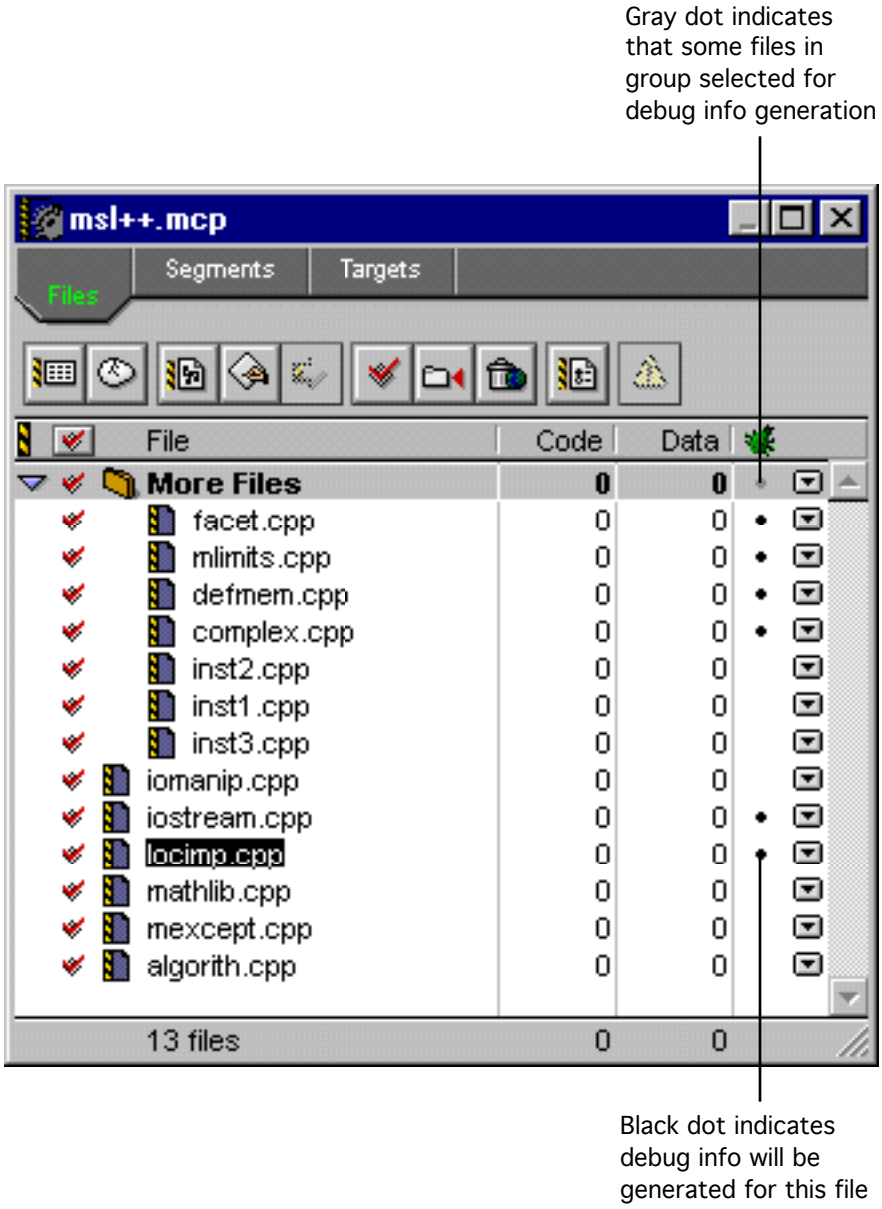
Activating Debugging for a File

To generate debugging information for a source code file, click in the Debug column and a Debug Info Marker appears in the column, as shown in Figure 3.26 on page 98. When you add files to your project, CodeWarrior automatically sets this marker unless you have deselected the appropriate option in the project's Linker panel.

You can also set Debug Info generation in the Project Inspector window. To learn how to do this, refer to "Guided Tour of the Project Window" on page 42. When files are added to the project, their

debug state is set by the state of the Enable Debugger menu command setting in the Project Menu.

Figure 3.26 Debug Info markers



The Debug Info Marker indicates that debugging information will be generated for this file when the project is built. Clicking the

marker removes it and causes the source code file to not have debugging information.

When a Debug Info Marker is selected for the first time, the file is also marked for compilation the next time you build your project. Whenever a Debug Info Marker is changed, the file will be marked for recompilation the next time your project is built.

You can generate Debug Info for all the files associated with the current build target. Shift/Option click in the Debug column to enable or disable generating debugger information for all of the files.

NOTE: Marking a source code file for debug inclusion does not mean that a debug file is created during linking. The linker settings panel for the target contains the options that enable CodeWarrior to create a debugging file.

Debug Info marker for groups

The Debug Info marker also appears on the right end of group rows (Figure 3.26), and can be toggled on and off by clicking. The Debug column, for groups, may show one of three markers:

- **Black marker:** all files in the group generate debugging information
- **Grey marker:** only some of the files in the group generate debugging information.
- **No marker:** means no debugging info for all files in group.

Adding Preprocessor Symbols to a Project

Sometimes you may want to add your own symbol definitions to your project so that they are automatically included at the beginning of each source code file when you build your project.

An example of this using the C or C++ language would be:

```
#define GLOBAL_DEBUG
```

Maybe you want to define this symbol when building development versions of your code, but want to undefine it before shipping your final product.

To do this, you would create a precompiled header and insert this symbol definition into the header. See "Using Precompiled or Pre-processed Headers" on page 329 for more information on how to do so.

To learn more about this topic, refer to the *Inside CodeWarrior: C/C++ Tools* and *Inside CodeWarrior: Pascal Tools* manual.

4

Working with Files

This chapter introduces the concepts behind working with files in the CodeWarrior IDE.

Working with Files Overview

In this chapter we discuss opening, creating, saving, closing, comparing, and printing files in the CodeWarrior environment.

To learn about editing files, refer to "Source Code Editor Overview" on page 129.

To learn about working with files using revision control systems, refer to "Version Control System Overview" on page 353.

The sections in this chapter are:

- Creating a New File
- Opening an Existing File
- Saving a File
- Closing a File
- Printing a File
- Reverting to a Previously-Saved File
- Comparing and Merging Files & Folders

Creating a New File

To create a new untitled window where source code or text may be entered, choose the **New** command from the File Menu.

After the new window appears, a text insertion point appears on the first line of the window. The CodeWarrior IDE places text that you type at this insertion point.

To learn more about text editing in the window you have just created, see "Source Code Editor Overview" on page 129.

Opening an Existing File

There are several ways to open a file with the CodeWarrior IDE. The methods discussed here are:

- Opening Files from the File Menu
- Opening Files from the Project Window
- Opening Files from an Editor Window
- Opening a Related File

NOTE: You cannot open libraries with the CodeWarrior Editor because of their binary format.

Opening Files from the File Menu

You can open two types of files:

- Project file—a file containing information on building a CodeWarrior project
- Text file—a source code, interface, or other text file

Project file

To open a project file, choose the **Open** command from the File Menu. For information on opening CodeWarrior project files, see "Opening an Existing Project" on page 61.

Text file

To open a text file or a source code file, choose the **Open** command from the File Menu. The IDE displays an Open dialog box, as shown

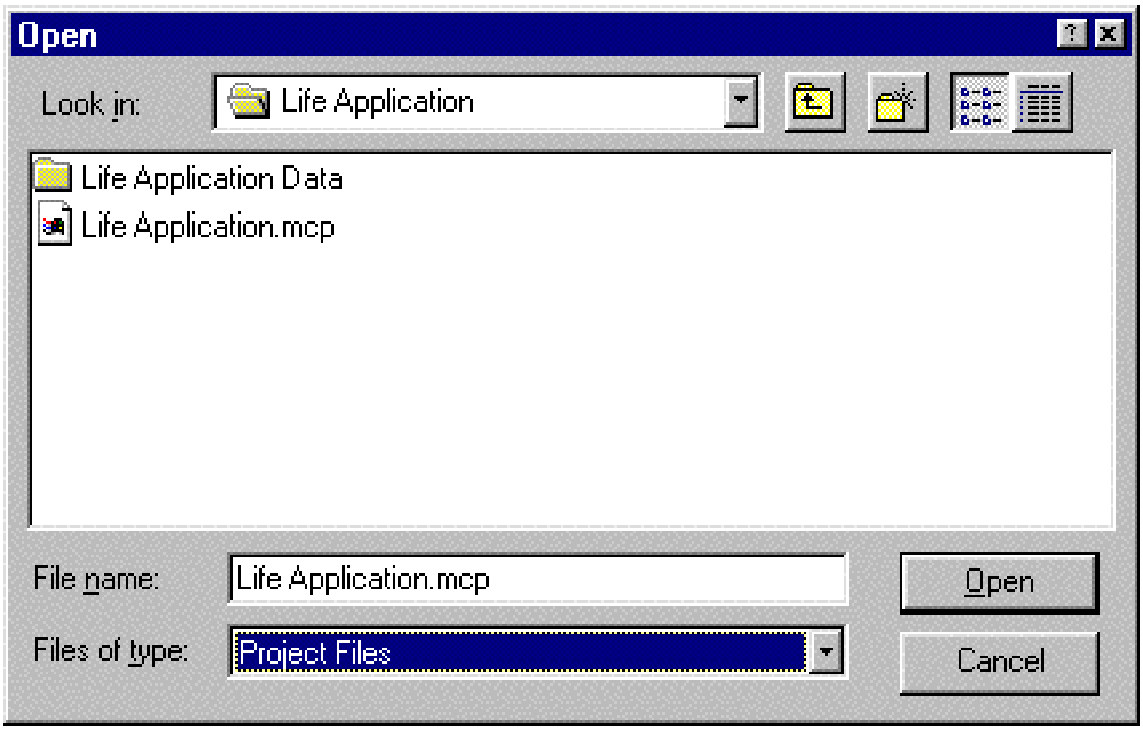
in Figure 4.1 (Windows), Figure 4.2 (Mac OS), and Figure 4.3 (Solaris).

Windows From the **Files of Type** pop-up menu, select **All Files**. The list of files changes to show all the files in the current folder, including text files.

Mac OS The dialog box displays a list of the available project and text files in the current folder. If the file is a stationery text file, the IDE will open a new, untitled Editor window and copy the contents of the stationery file into the window.

Solaris The dialog box displays a list of the available project and text files in the current directory.

Figure 4.1 Open dialog box (Windows)



Select the file you would like to open, and click Open. The CodeWarrior IDE opens the file in an Editor window.

For more information about editing source code, see "Source Code Editor Overview" on page 129.

Figure 4.2 Open dialog box (Mac OS)

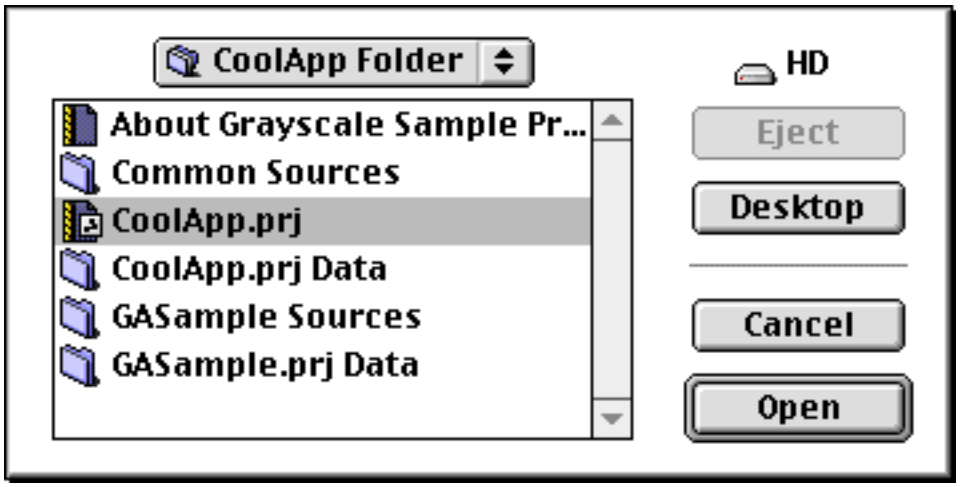
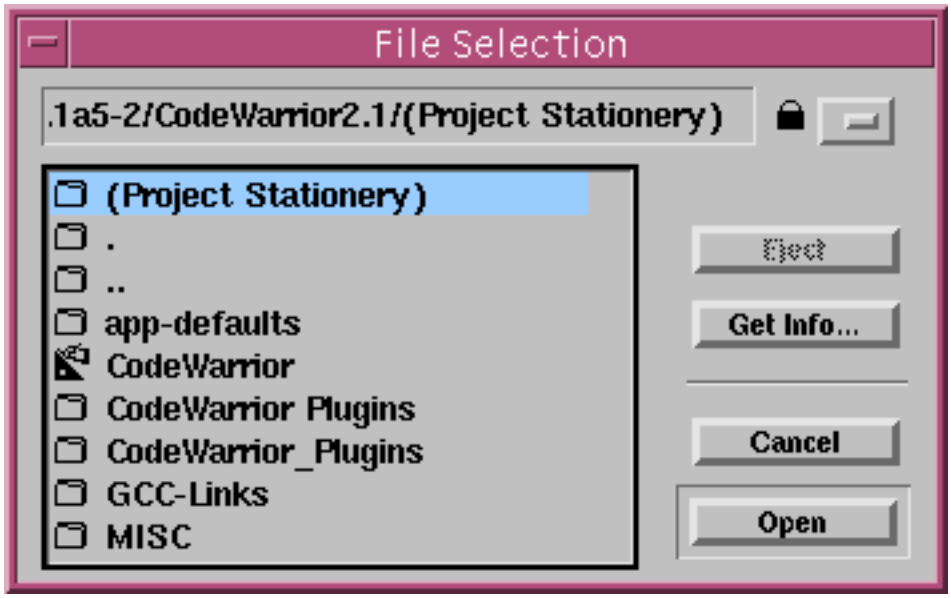


Figure 4.3 Open dialog box (Solaris)



Opening Files from the Project Window

There are different ways to open files from within the project window, depending on the type of file you wish to see. These different ways are:

- File column—opening a file that is in the project

- Group pop-up menu—opening a text source file from within a collapsed group
- Interfaces pop-up menu—opening an interface file included in a project's source file

---

TIP: (Mac OS) Files that contains binary data cannot be opened and displayed in a CodeWarrior Editor window. If the file was created using another application, double-clicking the file name in the File column will open the file in the application that created it. Refer to "Disassembling Source Code" on page 336 to learn how to view the contents of a library file.

File column

If the file you wish to see appears in the File column of the project window's File or Link Order views, double-click the file name to open it. If the file is a text file, CodeWarrior opens it in an Editor window.

Mac OS On the Mac OS, if the file is any other type, the CodeWarrior IDE opens the application that created the file. If the file is a binary file, such as a library file, it will not be opened, because binary files are not viewable using printable characters.

Another way to open a file is to select it, and then press the Enter/Return key. You can select multiple files in the Project window, and open them all by pressing the Enter/Return key. If you don't know how to select multiple files in a project, refer to "Selecting Files and Groups" on page 70.

For more information about the File Column, refer to "File column" on page 44.

Group pop-up menu

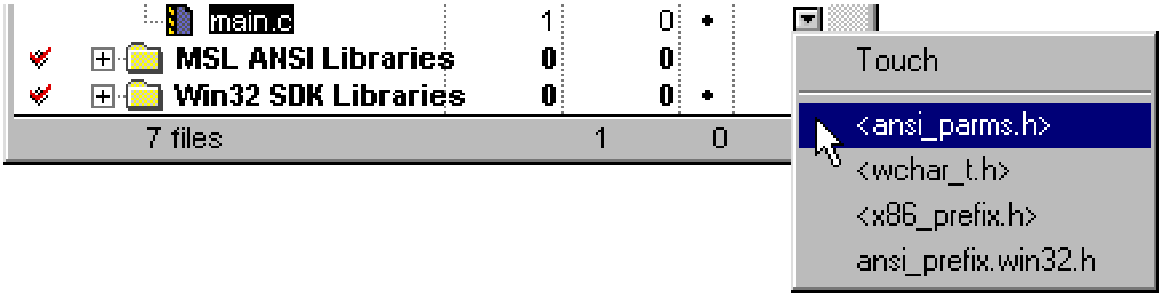
Another way to open a source file is to use the Interface pop-up for a group. A similar menu is shown in Figure 4.4. From this pop-up menu you may select the file in the group that you want to open.

You can open a source file by choosing it from the Interface pop-up menu for the group that contains the file. This works even if the group is collapsed and the file is not visible in the project window.

Interfaces pop-up menu

To open a header or interface file, click on the Interface pop-up to see a list of files. Select the file you want to open from this list, as shown in Figure 4.4.

Figure 4.4 Interfaces Files pop-up menu in the Project window



Note that header files inside "<...>" are system header files located within the Metrowerks CodeWarrior folder. Files without these symbols are header files that you have created. Your header files are stored in the same folder as your project or other Access Paths you have designated. To learn more about Access Paths and how to configure them, refer to "Access Paths" on page 298.

TIP: To switch between a source file and its interface file, use the same name for both files, except for the extension. For example, name your files `foo.cpp` and `foo.h`. Then press Ctrl/Command-` to instantly switch between the two files.

When the Interfaces File Pop-up is clicked for a library file that is part of your project, you will only have the option to Touch or Untouch the library file. Since libraries do not contain header or interface files, these files can not be opened from a pop-up corresponding to a library file.

To learn more about touching files, see "Touching and Untouching Files" on page 83.

Opening Files from an Editor Window

To open an interface file from within a source file you are editing, click the Interface Pop-Up Menu at the top left of the editor window as shown in Figure 5.2 on page 132. This pop-up menu lists all interface or header files used by the source file. Select a file from the Interface Pop-Up Menu to open it in a new editor window.

NOTE: If there are no files available in the menu, it means your text file does not contain source code, or that the source file has not yet been compiled.

Here's a different method. If you're editing any source code file, you can open an interface file mentioned anywhere in the text file with the Find and Open 'Filename' command.

First, select text in the editor window containing the name of the interface file you would like to open. An example of a file name you might see in a C source code file is `stdio.h`. You could select `stdio.h` by double-clicking on the `stdio` portion of the text. Then, choose the **Find and Open 'Filename'** command from the File Menu.

The CodeWarrior IDE then searches for the file and opens the file in an editor window. If the IDE can't find the selected file, a system beep sounds.

If you're editing a source code file and want to open a file without selecting any text, choose the **Find and Open File** command from the File Menu. This command will use the settings in the Access Paths for the project to search for the file to open.

After you choose Find and Open File, the CodeWarrior IDE then displays a dialog box, as shown in Figure 4.5. Type the name of the file you wish to search for in the Open editable text field.

Figure 4.5 Find and Open File dialog box

If you want to search both System Paths pane and User Paths pane directory paths (all paths specified in the Access Paths), turn the **Search Only in System Paths** option off.

To search only the CodeWarrior directory structure (the paths specified in the System Paths pane of the Access Paths), click on the **Search Only in System Paths** option to enable it.

To learn more about Access Paths and how to configure them, refer to "Access Paths" on page 298 for more information.

Opening a Related File

If you are working in a source code file and wish to open the corresponding interface file, or working with an interface file and wish to open the corresponding source file, use the keyboard shortcut Ctrl/Command-`. You can easily switch back and forth between the two files.

Saving a File

This section describes the many ways that the CodeWarrior IDE can save files. The topics discussed are:

- Saving one file
- Saving all files
- Saving files automatically

- Renaming and saving a file
- Backing up files
- Saving as a MS-DOS, Mac OS, or UNIX text file

Saving one file

To save your changes to the current Editor file, choose the **Save** command from the File Menu. The CodeWarrior IDE saves your file to your hard disk.

The Save command is dimmed if the window is new and has no data, if the contents of the active window have already been saved, or when the active window is the project window.

NOTE: If the file is new and untitled, the CodeWarrior IDE displays the Save As dialog box, described in "Renaming and saving a file" on page 110. Choose a name and location for your new file with this dialog box.

Projects are saved when they are closed, when you quit or exit the CodeWarrior IDE, or when the **Save A Copy As** command is selected. You don't need to explicitly save projects.

Saving all files

To save your changes to all the files currently open, press the keyboard shortcut Shift-Ctrl-S (Windows) or Option-Command-S (Mac OS). The CodeWarrior Editor saves all the modified files to your hard disk.

Saving files automatically

The CodeWarrior IDE can automatically save changes to all your modified files whenever you choose the **Preprocess**, **Precompile**, **Compile**, **Disassemble**, **Bring Up To Date**, **Make**, or **Run** commands from the Project Menu.

Using the Save All Before Build feature can save your work if your program should crash while running. However, if you're experi-

menting with a change and don't want to save changes, you may want to turn this option off.

To learn about how to enable or disable this feature, refer to the Save All Before Build option in the section of this manual titled "Editor Settings" on page 261.

Renaming and saving a file

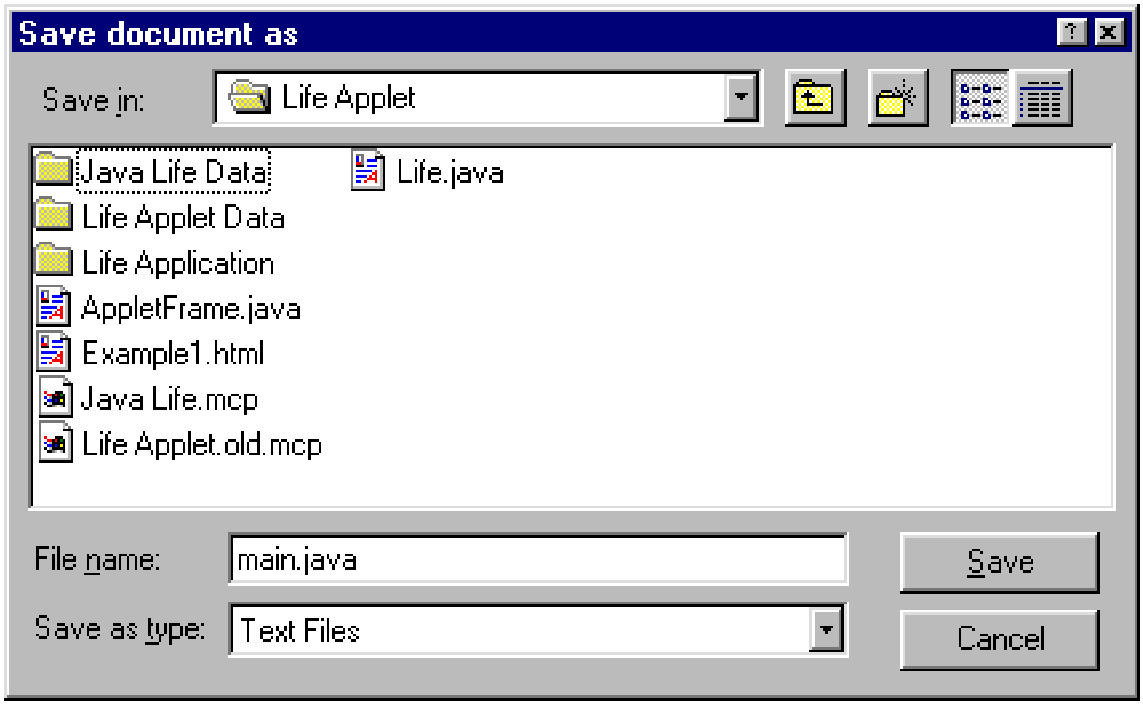
If you want to save a new untitled file or save a file under a new name, use the **Save As** command on the File Menu. If the file is in the current project, the CodeWarrior IDE updates the project to use the new name.

When you choose Save As from the File Menu, the CodeWarrior IDE displays the dialog box shown in Figure 4.6 (Windows), Figure 4.7 (Mac OS), or Figure 4.8 (Solaris).

Mac OS and Solaris Choose the Text or Stationery button to save the file as either a text file or a stationery file.

NOTE: (Mac OS) Beginning with Mac OS 8.5, the dialog box that you see when you select the Save As command will no longer appear as shown in Figure 4.7. Instead of clicking the Stationery radio button directly, you must first tell the IDE that you want to use the stationery option. To do this, select **Stationery Option** from the Format pop-up list. When the Stationery Option dialog box appears, click the Stationery radio button and click **OK**.

Figure 4.6 Save As dialog box (Windows)



Choose the file location and name the file, then click the **Save** button.

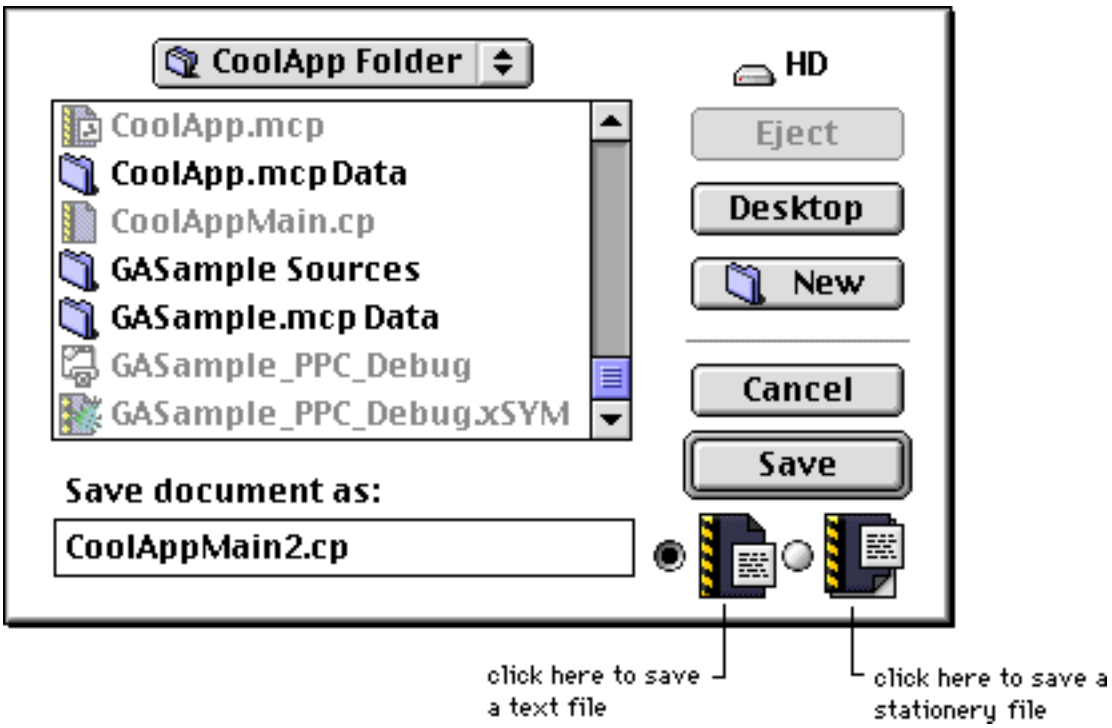
The CodeWarrior IDE saves the file and changes the name of the editor window to the name you entered.

If the file is in the current project, the CodeWarrior IDE changes the file's entry in the project to match the saved name. If you don't want to change the project, but still want to save the file, you can refer to the following section, "Backing up files."

Backing up files

If you want to save a backup copy of a text file before you make some changes to the original, use the **Save A Copy As** command in the File Menu. The CodeWarrior IDE creates a copy of the file under a new name that you specify, but leaves the original file unchanged and does not change the currently-open project to use the new file name.

Figure 4.7 Save As dialog box (Mac OS)



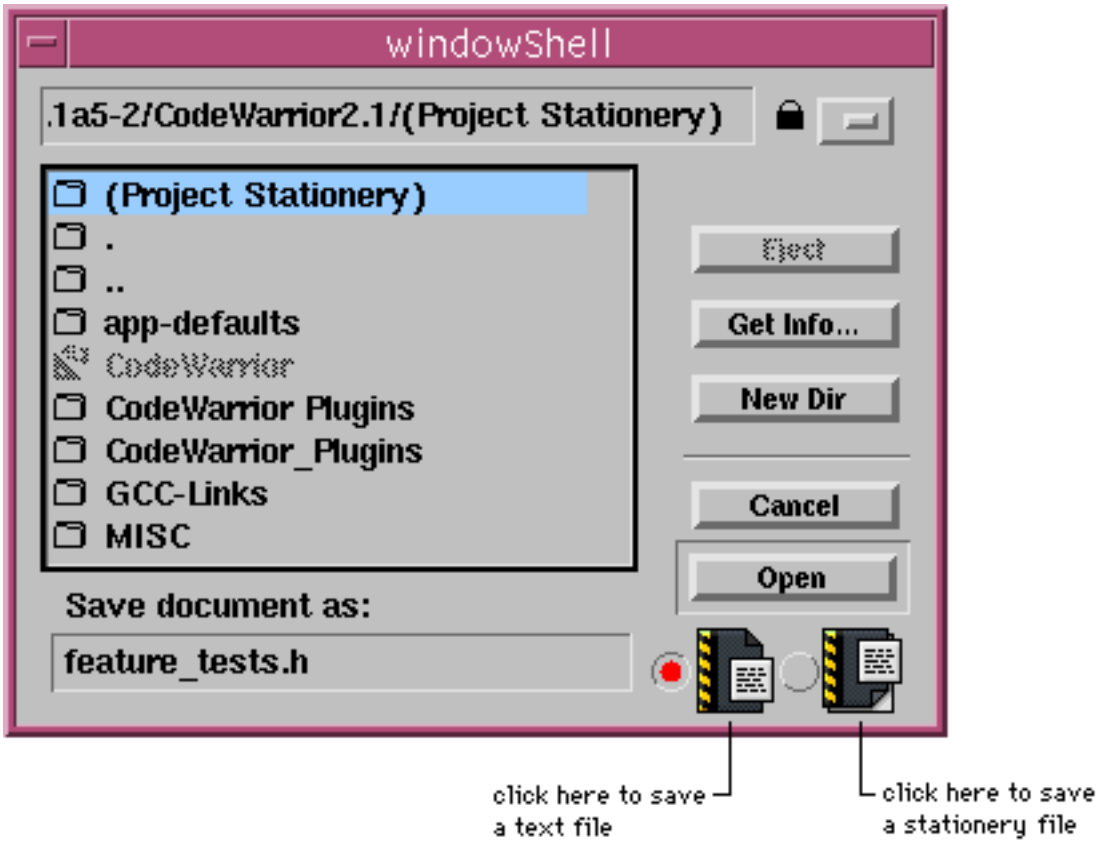
After choosing Save A Copy As from the File Menu, the CodeWarrior editor displays the dialog shown in Figure 4.6 (Windows), Figure 4.7 (Mac OS), or Figure 4.8 (Solaris). Specify the file's new location and choose a unique name for the file.

Mac OS and Solaris You can also choose whether to save the file as text stationery or a text file.

Now click **Save** and CodeWarrior saves a version of the file with your new name. It does not change the file in the editor window or in the current project.

If the project window is the active window, Save A Copy As allows you to save the project using a new name, or as a text file. You decide which type of project to create using the **Save Project As Type** pop-up menu shown in Figure 4.9. If you save the project as a text file, that text file will contain the names of all the files in the project.

Figure 4.8 Save As dialog box (Solaris)



Saving as a MS-DOS, Mac OS, or UNIX text file

When you open a text file originally created in a Windows, Mac OS, or UNIX text editor, CodeWarrior internally converts the text file to be compatible with the host platform and corrects inconsistent line endings. When you finish editing the file, CodeWarrior saves the file in its original format.

To learn about saving a text file under a different text format, see "Options Pop-Up Menu" on page 134.

Figure 4.9 Saving a copy of a project window (Windows)

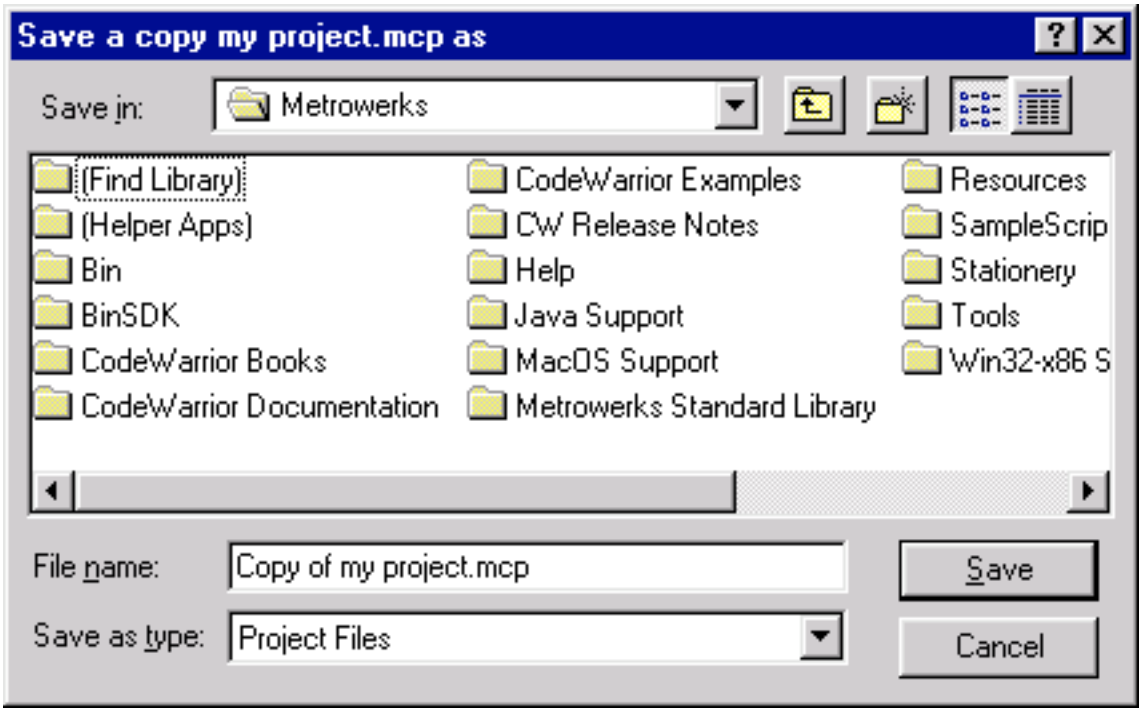


Figure 4.10 Saving a copy of a project window (Mac OS)

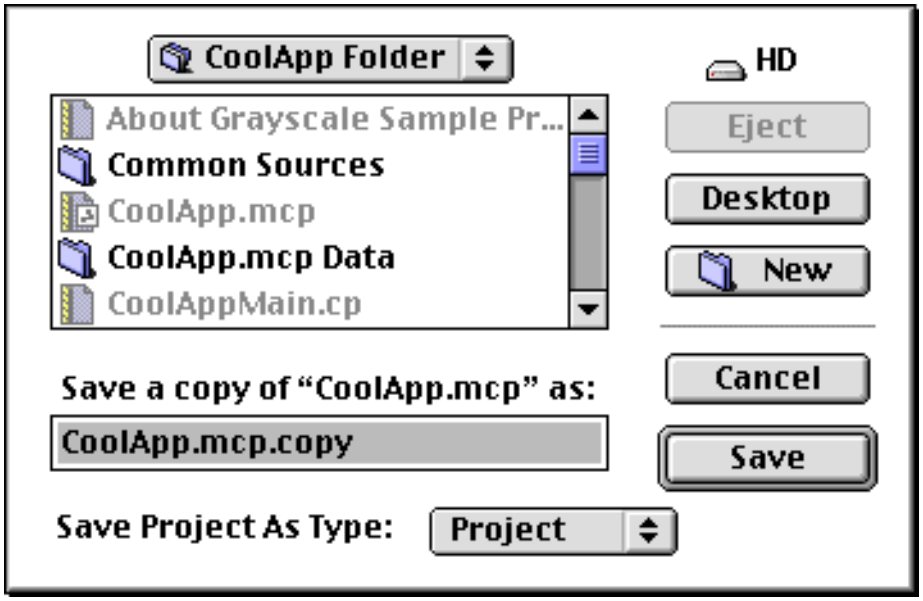
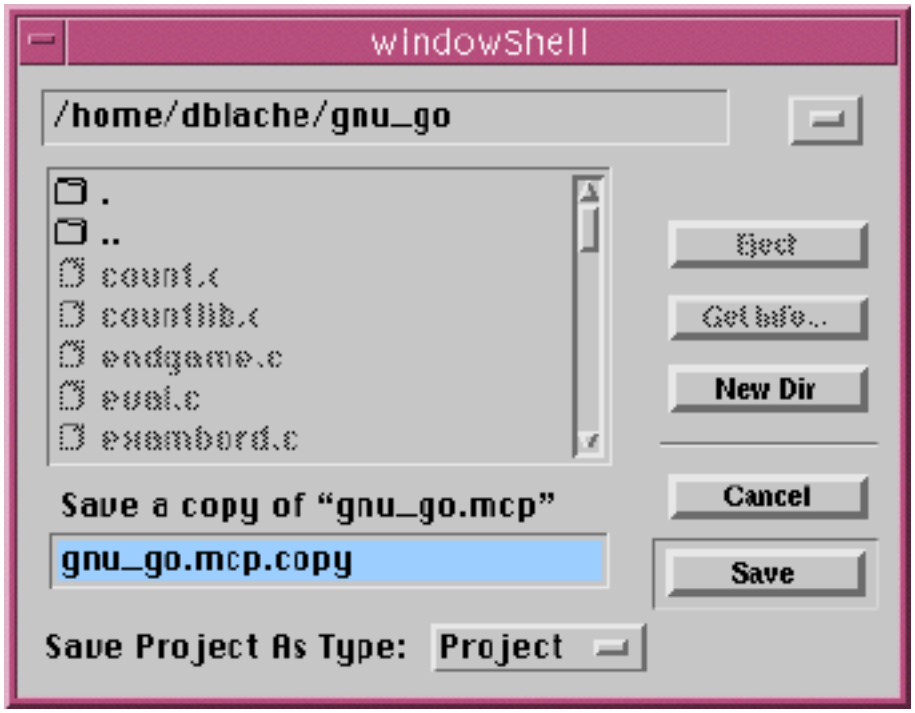


Figure 4.11 Saving a copy of a project window (Solaris)



Closing a File

Every editor or project window in the CodeWarrior IDE that you have opened is associated with a file on the hard disk. When you close the window, you close the file. You can close all windows or just a single CodeWarrior IDE window.

The topics in this section are:

- Closing One File
- Closing All Files

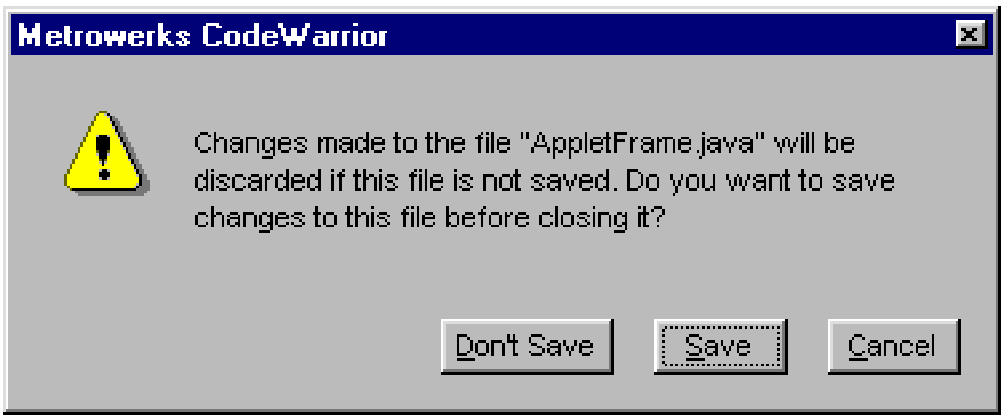
Closing One File

To close a window, choose **Close** from the File Menu.

If you close a text file using the File Menu and have not yet saved your changes, the CodeWarrior IDE asks if you want to save the

changes before closing the window, as shown in Figure 4.12. If you choose to close the file without saving your changes, all changes are lost.

Figure 4.12 The dialog box for unsaved changes


Metrowerks CodeWarrior
Changes made to the file "AppletFrame.java" will be discarded if this file is not saved. Do you want to save changes to this file before closing it?
Don't Save
Save
Cancel

Another way to close a window is by clicking the close box of the active window. This is exactly the same as choosing the Close command in the File Menu.

Closing an active project window automatically saves the project, and you will not see the dialog shown in Figure 4.12. For more on saving project files, consult "Saving a Project" on page 65.

The Close command also saves other properties of the window, such as the size, location, and the selected text in the active window. Refer to "Editor Settings" on page 261 for information on how to configure these options. If the appropriate options are enabled, the next time the source code file is opened, it will occupy the same position on your screen and the same text will be selected.

Mac OS The Close command saves the window position information in a format that is compatible with the Macintosh Programmer's Workshop (MPW). If you open a saved CodeWarrior Editor file with MPW, the same window parameters are respected. For more information about MPW, refer to the documentation on the CodeWarrior Reference CD.

Closing All Files

To close all open windows, use the **Close All** command. If you modified any file during the edit session, the Editor prompts you to save information before closing each window that contains changes.

Close All only closes editor and debugger windows. The Find dialog box, as well as project windows, remain open.

Windows This command is available by pressing Ctrl-Shift-W.

Mac OS This command is available when you press the Option key before clicking on the **File** menu.

TIP: To close all Editor windows at once, press Alt/Option and click on the close box of an Editor window.

Printing a File

Use the print options in the CodeWarrior IDE to print open files, a project file, or the contents of a window.

The topics in this section are:

- Setting Print Options
- Printing a Window

Setting Print Options

To configure printing options, choose the printer setup appropriate for your platform. CodeWarrior displays the appropriate printer dialog box. Choose Print Setup (Windows) or Page Setup (Mac OS) from the File Menu to display the printer dialog box.

Use this dialog box to select the paper size, orientation, and other settings. The specific settings and options depend on the printer you have connected to your computer. For more information on using your printer, consult the documentation packaged with your computer and printer.

If you Click **OK**, CodeWarrior saves the options for the next time you print any files.

Printing a Window

To print a window, make the window active and choose the **Print** command from the File Menu. This menu command allows you to print some or all of the active window.

When you choose this command, the CodeWarrior IDE displays the print dialog box for your printer. There are two additional CodeWarrior-specific options available for configuration in this dialog box. Depending on your printer and printer software, these options may be displayed in various places in different print dialog boxes.

The CodeWarrior-specific options are:

- Print Selection Only
- Print using Syntax Highlighting

Print Selection Only

If there is selected text in the Editor window you are printing, the **Print Selection Only** option appears. When this option is on, the CodeWarrior IDE prints only the selected text in the window, not the entire file. When this option is off, the CodeWarrior IDE prints the entire file.

Print using Syntax Highlighting

When the **Print using Syntax Highlighting** option is on, the CodeWarrior IDE prints the file with syntax coloring. On a black and white printer, the colors come out as shades of gray. When the Print using Syntax Highlighting option is off, the CodeWarrior IDE prints the file in black and white without syntax coloring.

TIP: (Mac OS) To print color syntax-highlighted text in **bold** and comments in *italics*, choose Black & White printing from the Print dialog box and turn on Print using Syntax Highlighting.

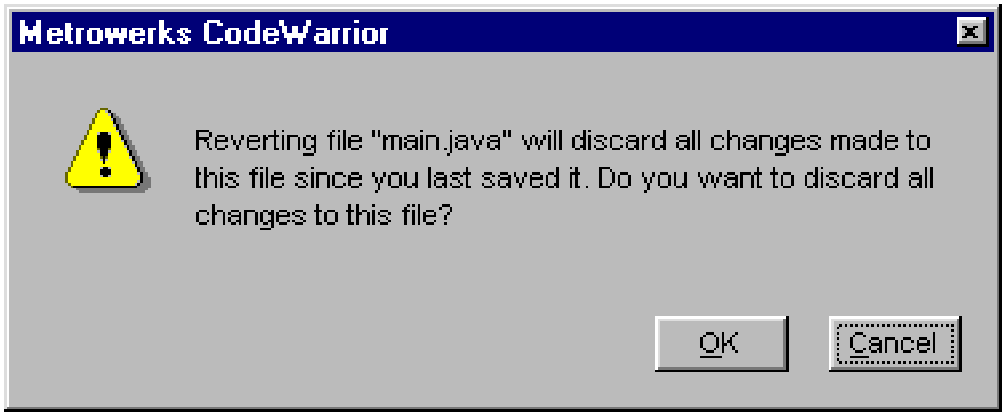
When you are finished configuring printing parameters, click the **Print** button in the printer dialog box. The CodeWarrior IDE then spools the file to your printing software for printing.

For information on other options in the print dialog box, see the documentation that came with your printer or printer software.

Reverting to a Previously-Saved File

If you've opened a text file and started editing it, then realize that you don't want to use the changes you've made, use the **Revert** command on the File Menu. When you select this command the dialog box shown in Figure 4.13 appears.

Figure 4.13 Revert to a Previous File


Metrowerks CodeWarrior
Reverting file "main.java" will discard all changes made to this file since you last saved it. Do you want to discard all changes to this file?
OK
Cancel

If you click the **OK** button, the last copy of the file you're working with will be opened, and all changes you have made since the last time you saved the file are lost. If you click **Cancel**, the file you're working with is not changed or saved to disk, and you can continue editing it.

Comparing and Merging Files & Folders

The CodeWarrior IDE has a facility to compare two text files, mark the differences between the files, and apply changes between the files. In addition, you can also compare the contents of two folders.

The topics in this section show you how to use the IDE's file comparison features:

- File Comparison and Merge Overview
- Choosing Files To Compare
- Examining and Applying Differences
- Choosing Folders To Compare

File Comparison and Merge Overview

The IDE's file comparison window displays two text files and the differences—insertions and deletions— between them. Figure 4.14 shows an example file comparison window. The window has controls to examine, add, and remove the differences between the files. The currently selected difference is shown with a darker color and outlined in black to contrast it from the other differences visible in the window.

The file comparison window has these parts:

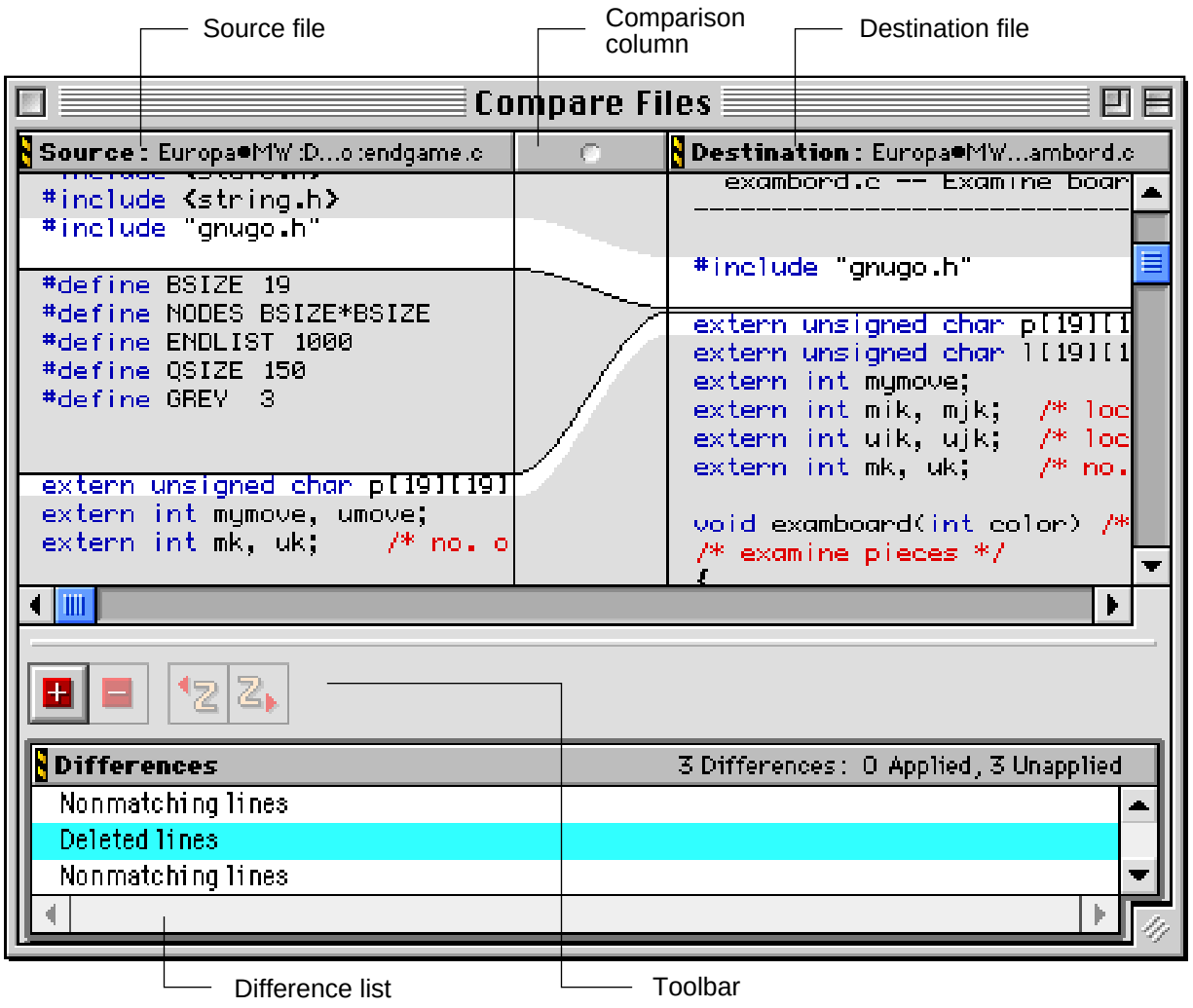
Source file

Displays the source text file that IDE uses as a basis for its comparison with the destination file. This pane appears on the left side of the file comparison window.

Destination file

Displays the source destination file that's compared with the source file. This pane appears on the right side of the file comparison window. Differences between the source file and the destination may be added to or removed from the destination file.

Figure 4.14 The file comparison window



Comparison column

Shows a graphical representation of where text was added or removed between the source and destination files. This column appears between the source and destination panes in the comparison window.

Difference list

Lists the insertions, deletions, and lines of mismatching text between the two files. Selecting an item in the list displays the difference in the source and destination panes. The comparison column also shows how and where the difference occurs between the two files. Text in the difference list will appear in *italics* when a difference is applied.

Toolbar

Has buttons to apply or remove changes between the two files to the destination file. The toolbar also has buttons to undo and redo changes to the source and destination files. For information on configuring toolbars, see "Customizing Toolbars" on page 280.

The controls include:

Control	Description
	Apply difference
	Unapply difference
	Undo
	Redo

Choosing Files To Compare

To open a file comparison window, choose **Compare Files** from the Search Menu to show a dialog box that prompts you for two files, source and destination files, to compare. To use a file dialog box to browse for the source and destination files, click their respective **Choose** buttons. You may also drag and drop text files into their respective boxes.

Windows Make sure that the **Compare Files** radio button, shown in Figure 4.15, is enabled. This allows you to select files by using the Choose buttons.

Text Compare Options

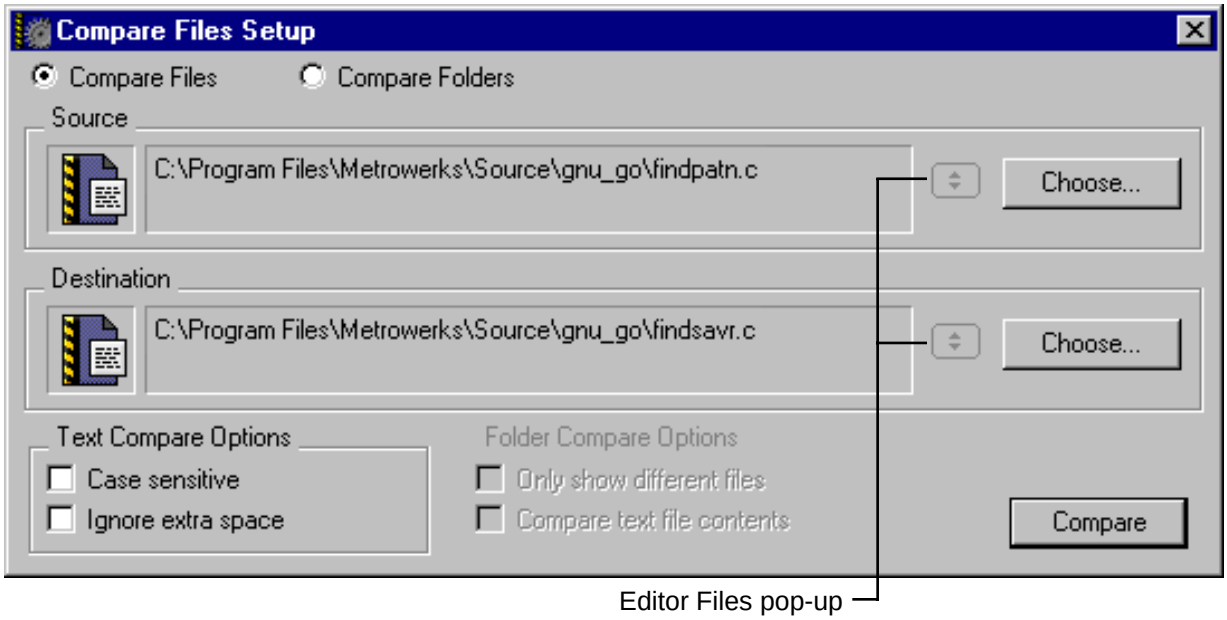
Select the **Case Sensitive** checkbox to consider the case of letters as part of the comparison operation. To ignore the case of letter, deselect this checkbox.

To take space and tab characters into account while comparing, select the **Ignore Extra Space** checkbox. To ignore extra space and tab characters, deselect this checkbox.

See "Folder Compare Options" on page 125 for information on the folder comparison options.

To show the comparison window after choosing files and setting options, click the **Compare** button.

Figure 4.15 Compare Files Setup dialog box



Examining and Applying Differences

Use the comparison window's toolbar and difference list to choose among the differences between the source and destination files and apply changes to the destination file.

To view a difference between the two files, click its entry in the difference list. To apply the difference, click the Apply button in the toolbar or choose **Apply Difference** from the Search Menu. To reverse a difference you've already applied, click the **Unapply** button or choose **Unapply Difference** from the Search Menu.

WARNING! Currently, using the Apply Difference and Unapply Difference buttons erase all actions from the Undo stack. In other words, when you exit the Difference window after apply or unapplying, all undo and redo actions will have been cleared from the Undo stack.

To Compare Editor Files

To compare two files that are already open in Editor windows, click on the **Editor Files** pop-up menu next to the source and destination paths, as shown in Figure 4.15. A list of open editor windows appears. Choose a file name from the menu to make it the source or destination file. Click **Compare** once the source and destination files are chosen to display the comparison window.

Choosing Folders To Compare

To open a folder comparison window, choose **Compare Files** from the Search Menu to show a dialog box that prompts you for two folders, the source and destination folders, to compare. To set these folders, drag and drop the folders into their respective boxes (Figure 4.16).

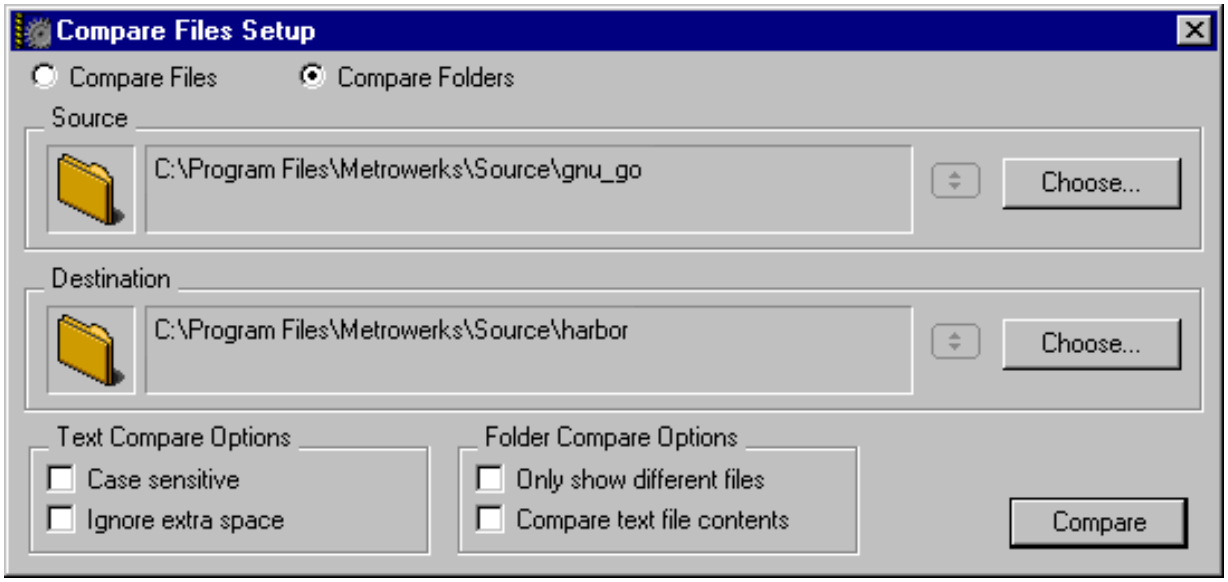
Windows To enable Folder Comparison Options as described in the following section, click the **Compare Folders** radio button. Clicking this radio button also allows you to select folders by using the Choose buttons in the Compare Files Setup dialog box.

Folder Compare Options

Select **Only Show Different Files** to only display files that are different in both folders in the Files In Both Folders list of the Compare Folders window (Figure 4.17). By default, this option is disabled, so all files in the source and destination folders are displayed.

Comparisons between files in the source and destination folders are normally based upon the file modification dates and file sizes. This is usually good enough to determine if there are differences between the two files. If there are invisible items in the folders, the comparison will skip over those items.

Figure 4.16 Compare Folders Setup dialog box



Select **Compare Text File Contents** to perform a more accurate compare of the files in the two folders. In essence, this performs a **Compare Files** command on every file in the source and destination folders and checks neither the modification dates nor the file sizes. This option is a lot slower since every file has to be opened, but the comparison information is more accurate.

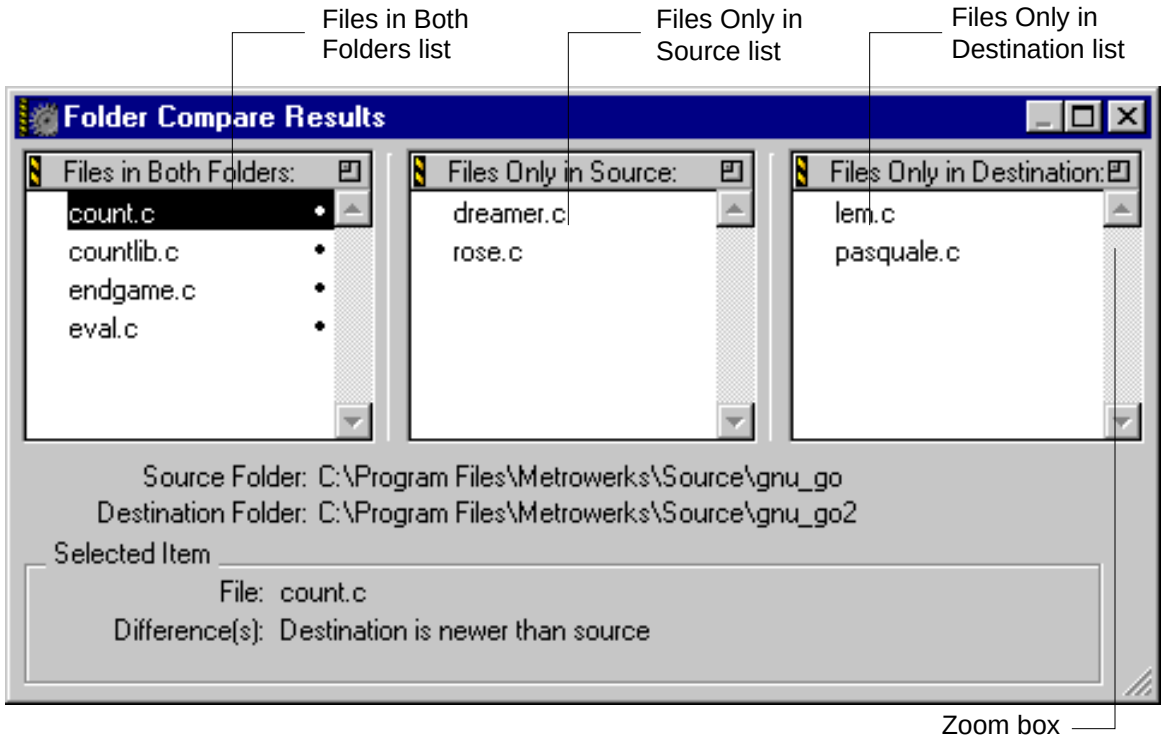
See "Text Compare Options" on page 123 for information on the file comparison options.

When you click the **Compare** button, the IDE displays the Folder Compare Results Window, as shown in Figure 4.17. Source code files, header files, text files, and folders appear in plain face. All other files appear italicized.

The Files In Both Folders list displays all files in both the source and destination folders unless the **Only Show Different Files** option is enabled. Files that are different in the two folders have a small bullet positioned to the right of their name.

When you click on a file in the Files In Both Folders list, Files Only In Source list, or Files Only In Destination list, specific information about the selected file appears in the Selected Item box at the bottom of the folder comparison window.

Figure 4.17 Folder Compare Results window



Double-click on a file in the Files In Both Folders list to open a Compare Files window for resolving the differences between the two differing files.

The Files Only In Source list displays all the files that appear only in the source folder while the Files Only In Destination list displays only files that appear within the destination folder.

You can click on a zoom box for any of the three lists to expand them to fill the window. Click again to collapse back to their original size.

5

Editing Source Code

This chapter explains how to use the CodeWarrior IDE text editor to edit your source code.

Source Code Editor Overview

The CodeWarrior Editor is a full-featured text editor specially designed for programmers, with features such as:

- Pop-up menus on every editor window for opening your interface files and navigating among your routines quickly.
- Syntax highlighting that formats source code for easy identification of comments and keywords in your source files.
- Convenient on-line reference material for routines available instantly. You just point and click.

The topics in this chapter are:

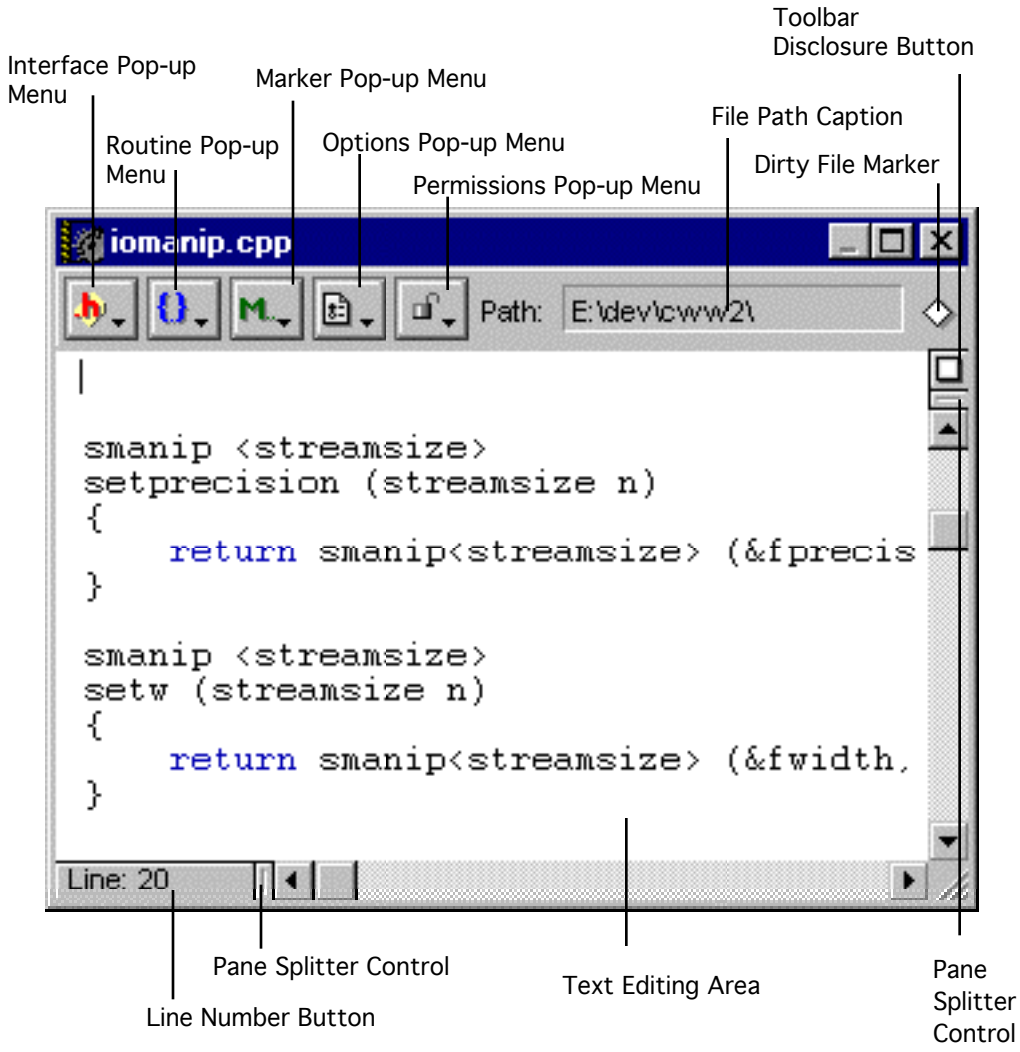
- Guided Tour of the Editor Window
- Editor Window Configuration
- Basic Text Editing
- Navigating the Text
- Online References

You can also customize options that affect the way the CodeWarrior Editor works. To learn more about how to do this, refer to "Editor Settings" on page 261.

Guided Tour of the Editor Window

The CodeWarrior Editor window, shown in Figure 5.1, contains elements you'll find useful when viewing and editing your source files.

Figure 5.1 The Editor window



To see an editor window, create a new text file using the **New** command on the File Menu.

The sections that follow describe the elements of the editor window shown in Figure 5.1.

- Text Editing Area
- Interface Pop-Up Menu
- Routine Pop-Up Menu
- Marker Pop-Up Menu
- Options Pop-Up Menu
- VCS Pop-Up Menu
- File Path Caption
- Dirty File Marker
- Pane Splitter Controls
- Line Number Button
- Toolbar Disclosure Button
- Path Pop-Up Menu (Mac OS)

Text Editing Area

The Text Editing Area of the editor window is where your text is entered and edited.

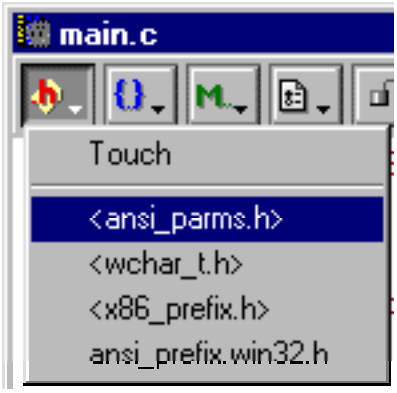
You may select and drag text out of an editor window to any destination that can accept a drop, such as another open editor window. You may also drag selected text into an editor window from other applications that support drag-and-drop.

For more information about drag and drop operations with text, see "Moving Text (Drag and Drop)" on page 146.

Interface Pop-Up Menu

Use the Interface pop-up menu shown in Figure 5.2 to open interface or header files referenced by the current file. You can also use the Touch and Untouch commands from this pop-up.

Figure 5.2 The Interface pop-up menu



To open a file in the list, scroll down to the file you'd like to see and release the mouse button. Note that in order to see a list of files in the menu, the project file must be opened. Note also that some files cannot be opened, such as precompiled header files and libraries.

For more information on opening files, see "Opening an Existing File" on page 102.

To recompile your file the next time the project is built, you choose the **Touch** command. If you click on the Interface pop-up again you can deselect the file for compilation with the **Untouch** command in the menu.

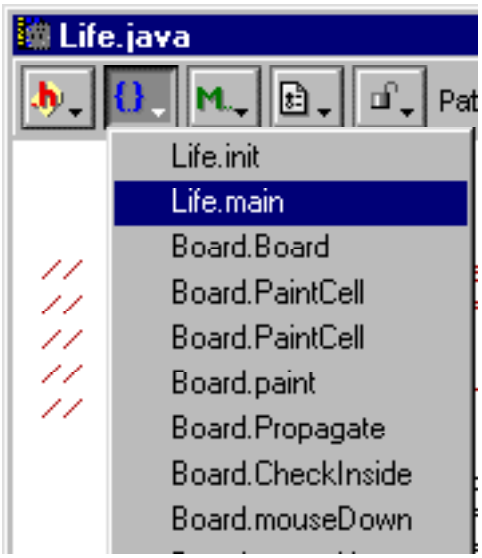
To learn more about touching files, see "Touching and Untouching Files" on page 83.

Routine Pop-Up Menu

Use the Routine pop-up menu shown in Figure 5.3 to set the current location of the text insertion point in your text files.

The Routine pop-up menu lists the routines in your source file. The highlighted or bulleted routine in the pop-up tells you where the text insertion point is currently located.

Figure 5.3 The Routine pop-up menu



NOTE: If the pop-up is empty, the file is not a source file.

NOTE: By default, the menu lists the routines in the order in which they appear in the file. If you'd like to list routines alphabetically, hold down Ctrl/Option and click on the routine icon.

If you'd like to change the default display order of the routines to alphabetical, enable the Sort Function Popup option. See "Editor Settings" on page 261 for more information about editor options.

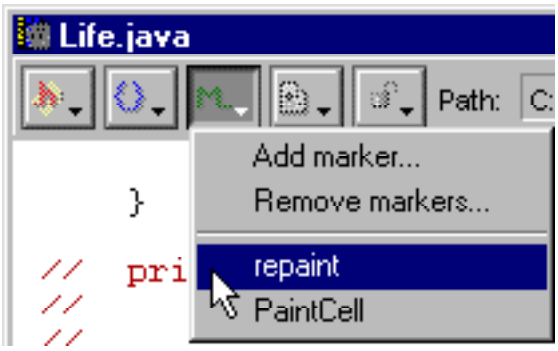
TIP: If you're editing a Pascal file, the Routine pop-up menu displays function names in *italics*, procedure names are in plain face, and the main program is in **bold**.

Marker Pop-Up Menu

Use the Marker pop-up menu shown in Figure 5.4 to add and remove markers in your text files. These markers are

easy to use and convenient for quick access to a line of text, remembering where you left off, and other identification purposes.

Figure 5.4 The Marker pop-up menu



For more information on using markers, see "Adding, Removing, and Selecting a Marker" on page 151.

Options Pop-Up Menu

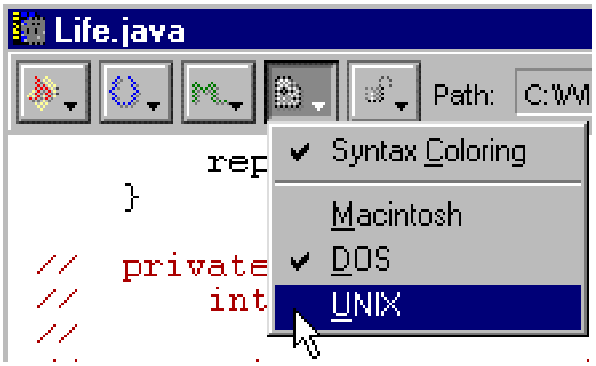
Use the Options pop-up menu, shown in Figure 5.5, to turn color syntax highlighting on or off for the current file, and also to set the format for how to save the file.

To enable or disable syntax highlighting, choose **Syntax Coloring** from the Options pop-up menu.

The type of file currently open in the Editor window is indicated by a check mark next to the **Macintosh**, **DOS**, or **UNIX** options in the pop-up menu. The next time you save the file, the CodeWarrior IDE saves it in the format you select.

For more information on the Syntax Coloring option shown in this menu, see "Syntax Coloring" on page 266.

Figure 5.5 The Options pop-up menu



VCS Pop-Up Menu

The VCS pop-up menu indicates the read/write revision control database status of the current file. If the pop-up icon box shows an Unlocked icon or the Read/Write icon, you can modify the file you're working with. The icons and their meanings are described in the section called "Using Source Code Control with Files" on page 359.

Using this pop-up menu, you can get a new copy of your file, checkout the file for modification, make it writable so you can make changes without doing a checkout, and other operations.

For more information about revision control system software, see "Using Source Code Control with Files" on page 359.

File Path Caption

The CodeWarrior IDE automatically displays the directory path of the current file in the File Path Caption, which is at the top right of the window shown in Figure 5.1 on page 130.

Mac OS To learn about another method of determining the path of a file, refer to "Path Pop-Up Menu (Mac OS)" on page 137.

Dirty File Marker

The Dirty File Marker tells you if the file displayed in a window has been modified since it was last saved or opened. The states of the Dirty File marker include:

Unchanged file

Modified and unsaved file (“dirty”)

Pane Splitter Controls

Pane Splitter Controls split the editor windows into panes so you can view different portions of a file in the same window.

You use these controls to adjust the sizes of the panes after you’ve created them. Figure 5.8 on page 139 shows an editor window with multiple panes.

For more information on this topic, see “Splitting the Window into Panes” on page 139.

Line Number Button

The line number box shown in Figure 5.1 displays the number of the line that contains the text insertion point. You can also use this button to go to another line in the file.

For information about setting the text insertion point on another line, see “Going to a Particular Line” on page 154.

Toolbar Disclosure Button

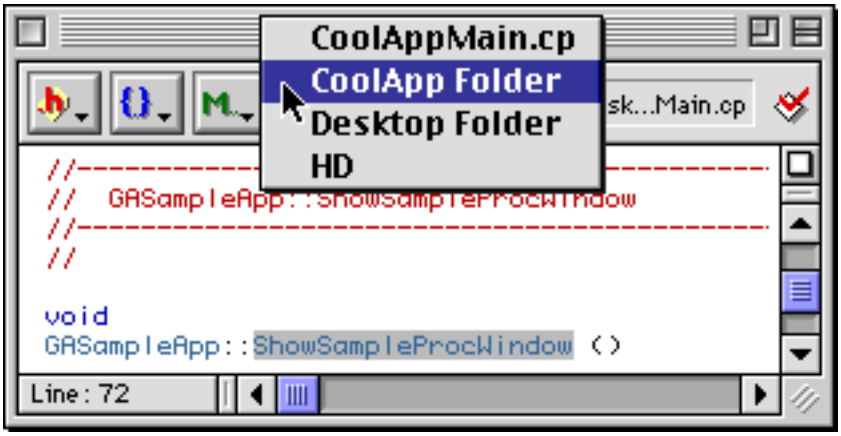
The Toolbar Disclosure Button hides or displays the Editor window’s toolbar along the top of the window. If the toolbar is hidden, a row of smaller controls appears at the bottom of the Editor window (see Figure 5.7 on page 138).

For more information on using the Toolbar Disclosure Button, refer to"Seeing Window Controls" on page 138.

Path Pop-Up Menu (Mac OS)

To see the directory path of the file in the active Editor window, press the Command key and click on the name of the file in the title bar of the window, as shown in Figure 5.6. You can also directly open a folder by choosing it from the menu.

Figure 5.6 File Path pop-up in the window titlebar



Editor Window Configuration

The editor allows you to customize your view of the file you're working with. In this section, you'll learn about the following topics:

- Setting Text Size and Font
- Seeing Window Controls
- Splitting the Window into Panes
- Saving Editor Window Settings

To learn about configuring the Editor window's toolbar, see "Customizing Toolbars" on page 280.

Setting Text Size and Font

You use the Font & Tabs preference panel to set the size or font used to display text in an editor window. For more information on this topic, see "Font & Tabs" on page 265.

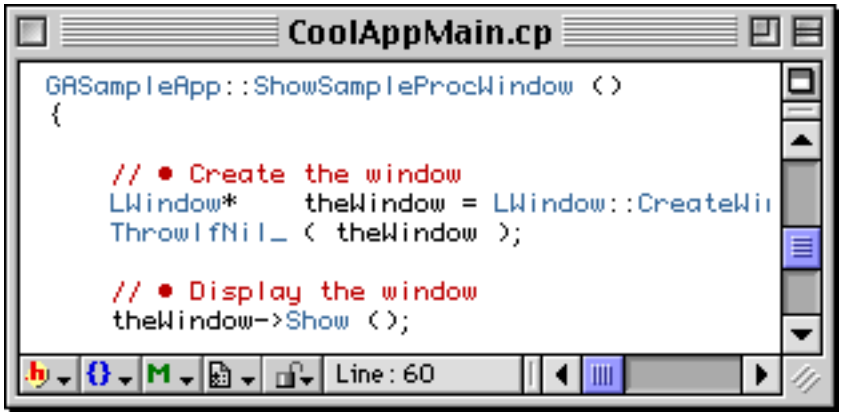
Seeing Window Controls

The row of pop-up menus and controls that appears along the top of the editor window is called the toolbar. The Toolbar Disclosure Button, shown in Figure 5.1 on page 130, is used to show or hide the toolbar.

However, if you hide the toolbar, the default pop-up menu controls appear along the bottom of the editor window, as shown in Figure 5.7. Note that the File Path Caption is no longer visible.

NOTE: If you hide a customized Editor window toolbar, custom items do not appear at the bottom of the window. The default items always appear when the toolbar is hidden. When you show the toolbar again, it retains its custom configuration. For information on toolbars in general, and customizing them, see "Customizing Toolbars" on page 280.

Figure 5.7 Pop-ups along the Editor window bottom



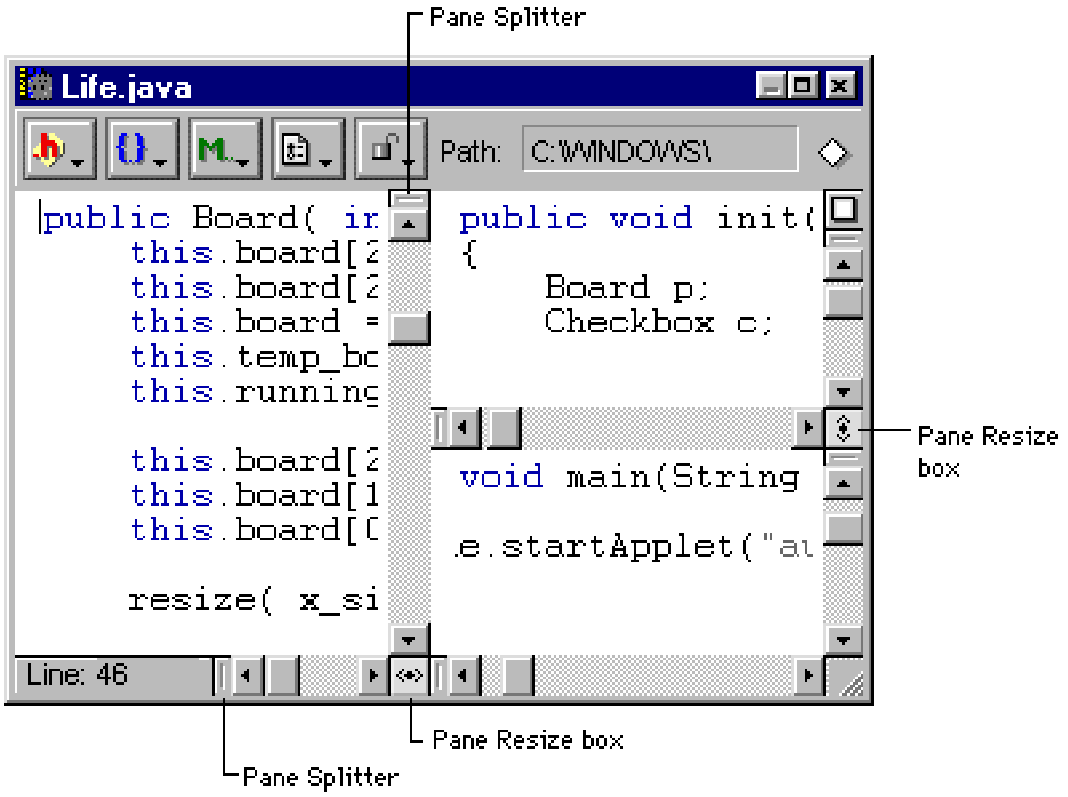
To show the toolbar along the top of the editor window again, click the Toolbar Disclosure Button once more.

You can choose a default setting to display or hide the toolbar in editor windows. To do so, see "Toolbar Disclosure Button" on page 136 for more information. See also "Showing and Hiding a Toolbar" on page 282.

Splitting the Window into Panes

You can split the editor window into panes to view different parts of a file in the same window, as shown in Figure 5.8. This section describes creating, adjusting, and removing multiple panes.

Figure 5.8 Multiple panes in a window



Creating a new pane

To create a new pane in an editor window, click and drag a Splitter Bar. Splitter bars are on each scroll bar of a pane in the editor window, on the top and left sides.

As you drag a Splitter Bar, a gray focus line tracks your progress and indicates where the new pane will go. When you release the mouse button, the editor creates a new pane.

You can also double-click the Splitter Bar to split a pane into two equal parts.

Resizing a pane

To change the sizes of the panes in an editor window, click and drag the pane resize boxes.

As you drag a resize box, a gray focus line indicates your progress. When you release the mouse button, the editor redraws the panes in their new positions.

Removing a pane

To remove a pane from an editor window, click and drag a Resize Box all the way to an edge of the window.

As you drag the Resize Box, a gray focus line indicates your progress. If you drag close to the edge of the window, the gray lines are no longer displayed. If you release the mouse button at that time, the editor removes one of the panes from the window.

You can also double-click on a Resize Box to remove a split.

Saving Editor Window Settings

The current settings of an editor window are automatically saved whenever the window is closed, or when the toolbar is hidden or shown.

The settings saved are the size and location of the window, and the setting of the Toolbar Disclosure Button. Any new editor windows you open will have these new default settings. Any windows you presently have open will need to be closed and reopened to get the new settings.

WARNING! (Mac OS) If you are using a revision control system for your source code files that uses native file locking, you must have your file checked out in order for the settings to be saved. If you are using `ckid` resources to lock the files, the information is saved for you without checking out the file.

To learn more about configuring editor window settings, refer to "Font & Tabs" on page 265.

To learn about saving default settings for CodeWarrior Browser windows, see "Saving a Default Browser" on page 233.

Basic Text Editing

The CodeWarrior IDE gives you lots of help in editing source files, all of it described in the topics that follow.

The topics in this section are:

- Basic Editor Window Navigation
- Adding Text
- Deleting Text
- Selecting Text
- Moving Text (Drag and Drop)
- Using Cut, Copy, Paste, and Clear
- Balancing Punctuation
- Shifting Text Left and Right
- Undoing Changes
- Controlling Color

Basic Editor Window Navigation

The CodeWarrior IDE gives you several ways to move the text insertion point in a file. Review this section again after you become more familiar with the CodeWarrior editor's features.

Scrollbar navigation

Like any other text editor, you may adjust the text to view in an Editor window by using the scroll bars.

The CodeWarrior IDE lets you configure how the scroll bars affect the window view when you drag the scroll bar thumb around. You can modify the way that scrolling behaves in the Editor windows of your project by changing the Dynamic Scrolling option. To learn how to do this, refer to "Dynamic Scrolling" on page 263.

Keyboard navigation

Table 5.1 describes how to move the insertion point around in a file with function keys.

For information on Solaris modifier key settings, see "Special Note for Solaris Users" on page 21.

Table 5.1 Text navigation with the keyboard

To move insertion point to	Windows	Mac OS	Solaris
Previous word	Control-Left Arrow	Option-Left Arrow	Alt-Left Arrow
Next word	Control-Right Arrow	Option-Right Arrow	Alt-Right Arrow
Beginning of line	Home	Command-Left Arrow	Meta-Left Arrow
End of line	End	Command-Right Arrow	Meta-Right Arrow

To move insertion point to	Windows	Mac OS	Solaris
Beginning of file	Control-Home	Option-Up Arrow	Alt-Up Arrow
End of file	Control-End	Option-Down Arrow	Alt-Down Arrow

Table 5.2 describes how to scroll to different locations in a file, without moving the insertion point. Note that some of the keys listed in the table may not be on your keyboard, depending on what kind of keyboard you have.

For information on Solaris modifier key settings, see "Special Note for Solaris Users" on page 21.

Table 5.2 Scroll with the keyboard

To move insertion point to	Windows	Mac OS	Solaris
Previous page	Page Up	Page Up	Page Up
Next page	Page Down	Page Down	Page Down
Beginning of file	Control-Home	Home	Home
End of file	Control-End	End	End
Previous line	Control-Up Arrow	Control-Up Arrow	Control-Up Arrow
Next line	Control-Down Arrow	Control-Down Arrow	Control-Down Arrow

Adding Text

To add text to a file you've opened, click once in the Text Editing Area of the window to set the new location of the text insertion point. After you see the insertion point at the new location, you may begin typing on the keyboard to enter text.

To read about different ways to move the insertion point in an Editor window, see "Basic Editor Window Navigation" on page 142.

Deleting Text

There are several different methods for deleting text.

To delete text that you just typed, press the Backspace/Delete key.

To delete text that is in front of the text insertion point, use the Delete/Del (⌦) key.

To delete more than one contiguous character at a time, select the text you want to delete and press the Backspace/Delete keyboard shortcut.

Mac OS To delete text from the text insertion point to the end of the file, use the Control/Command Delete keyboard shortcut.

If you don't know how to select text, read "Selecting Text" in the next section.

Selecting Text

There are several different ways to select text in the editor window.

Select text by holding down the Shift key while pressing any of the shortcuts listed in Table 5.1. You may also select a word, a line, or a range of text.

To select a word:

- double-click on the word

To select a line:

- triple-click anywhere in the line
- move the mouse pointer to the left edge of the editor window so that the mouse pointer points right and press the mouse button

This selection method is available when the **Left Margin Click Selects Line** option is on in the Editor Settings preference panel.

To select a range of text:

- click and drag the mouse in a portion of your window where there is text
- set your text insertion point to mark the beginning of your selection then press the Shift key while clicking the place in your text where you want the selection to end
- move the mouse pointer to the left edge of the editor window so that the mouse pointer points right, then click and drag the mouse pointer to select lines of text

  This selection method is available when the **Left Margin Click Selects Line** option is on in the Editor Settings preference panel.

Mac OS and Solaris The editor can select parts of text identifiers by holding down the Control key while using the left or right arrow keys, or when double-clicking. For example, double-clicking between the two "m" characters in `FindCommandStatus()` would result in the word `Command` being selected.

To display and highlight an entire routine in the editor window, press the Shift key while selecting a routine in the Routine Pop-Up Menu. This is particularly useful for copy and paste operations and for using drag and drop to move code around in your file.

Table 5.3 describes how to select text using the keyboard, starting at the current insertion point. Note that some of the keys listed in the table may not be on your keyboard, depending on what kind of keyboard you have.

For information on Solaris modifier key settings, see "Special Note for Solaris Users" on page 21.

Table 5.3 Text selection with the keyboard

To select text to	Windows	Mac OS	Solaris
Previous word	Shift-Control-Left Arrow	Shift-Option-Left Arrow	Shift-Option-Left Arrow
Next word	Shift-Control-Right Arrow	Shift-Option-Right Arrow	Shift-Option-Right Arrow
Beginning of line	Shift-Home	Shift-Command-Left Arrow	Shift-Command-Left Arrow
End of line	Shift-End	Shift-Command-Right Arrow	Shift-Command-Right Arrow
Beginning of page	Shift-Page Up	Shift-Option-Up Arrow	Shift-Option-Up Arrow
End of page	Shift-Page Down	Shift-Option-Down Arrow	Shift-Alt-Down Arrow
Beginning of file	Shift-Control-Home	Shift-Command-Up Arrow	Shift-Meta-Up Arrow
End of file	Shift-Control-End	Shift-Command-Down Arrow	Shift-Meta-Down Arrow

For more information about using drag and drop with text, see "Moving Text (Drag and Drop)" in the next section.

You can also select blocks of code quickly using the Balance command. To learn how to do this, refer to "Balancing Punctuation" on page 147.

Moving Text (Drag and Drop)

If you have some text in your editor window that you would like to move to a new location, you can use the drag and drop features of the editor to do it. In order to use drag-and-drop editing, this feature must be enabled in the IDE. To learn more about turning this feature on or off, refer to "Editor Settings" on page 261.

The CodeWarrior editor can also accept drag-and-drop text items from other applications that support drag and drop. To see if one of your applications supports drag and drop, refer to the documentation that came with it.

TIP: (Mac OS) If you press the Option key and drag selected text, the text will be copied to your new location instead of moved.

Using Cut, Copy, Paste, and Clear

There are standard menu commands available on most computer applications, called **Cut**, **Copy**, **Paste**, and **Clear**. In the CodeWarrior IDE, these commands appear on the Edit Menu.

Use these commands to remove text, or to copy and paste in a window, between windows, or between applications.

For more information about these commands, refer to "Edit Menu" on page 379.

Balancing Punctuation

When you're editing source code, you often have to make sure that every parenthesis (`( )`), bracket (`[ ]`), and brace (`{}`) has a mate, or the compiler could misinterpret your code or give you an error.

The CodeWarrior IDE provides several checks that help you balance these elements correctly.

To check for balanced parentheses, brackets, or braces, place the insertion point in the text you want to check. Then, choose **Balance** from the Edit Menu. Alternatively, double-click on a parenthesis, bracket, or brace character that you want to check for a matching character.

The CodeWarrior editor searches starting from the text insertion point until it finds a parenthesis, bracket, or brace, then it searches in the opposite direction until it finds the matching half. When it finds the match, it selects the text between them. If the insertion

point isn't enclosed or if the punctuation is unbalanced, the computer beeps.

TIP: Use the Balance command to select blocks of code quickly.

Using automatic balancing

You can have the editor check for balanced punctuation automatically. If you would like to learn more about checking the balance of code automatically as you type, refer to "Balance While Typing" on page 263.

Shifting Text Left and Right

Use the Shift Left and Shift Right commands on the Edit Menu to shift a block of text to the left or right.

To shift blocks of text, select a block of text. If you don't know how to do this, refer to "Selecting Text" on page 144. After selecting, choose **Shift Right** or **Shift Left** from the Edit Menu.

The CodeWarrior editor shifts the selected text one tab stop to the right or left by inserting or deleting a tab at the beginning of every line in the selection.

To learn more about controlling the number of spaces the text is indented, refer to "Font & Tabs" on page 265.

Undoing Changes

The CodeWarrior editor supplies ways to Undo mistakes as you edit a file.

Undoing the last edit

The Undo command reverses the effect of your last action. The name of the Undo command on the Edit Menu varies depending on your last action. For example, if you just typed in some text, the

command changes to Undo Typing. Choose **Undo Typing** to remove the text you just typed.

Undoing and redoing multiple edits

When the Use Multiple Undo option is on, you can Undo, Redo, Multiple Undo, and Multiple Redo previous actions by continuing to choose the Undo or Redo commands.

For instance, if you Cut a word, then Paste it, then type some text, you can backtrack all those actions by choosing **Undo** three times. The first Undo removes the text you typed, the second Undo unpastes the text you pasted, and the third Undo uncuts the text you Cut, therefore returning the text to its original condition.

You can perform those activities again in the same order by choosing the **Redo** command three times.

To enable the Use Multiple Undo option, refer to "Use Multiple Undo" on page 264.

Note that the keyboard shortcut for the Redo command changes when the Use Multiple Undo option is turned off.

WARNING! Undo actions are saved in a stack, therefore it's possible to lose actions when performing several undo and redo actions. Each undo action adds an item to the stack, while each redo repositions a pointer to the next undo action. For example, if there were five undo actions on the stack (ABCDE), and you redo two of them, the stack appears to the pointer like this: ABC. When you perform a new action (ABCF), the undo events (DE) are no longer available.

Reverting to the last saved version of a file

The **Revert** command on the File Menu returns a file to its last saved version. To learn more about how to revert to the previous version of a file, see "Reverting to a Previously-Saved File" on page 119.

Controlling Color

You can use color to highlight many elements in your source code, such as comments, keywords, and quoted character strings. Highlighting these elements helps you identify them in the text, so you can check your spelling and syntax as you type by recognizing color patterns. For information on configuring color syntax options, see "Syntax Coloring" on page 266.

You can also highlight custom keywords, which are in a list of words you designate. See "Syntax Coloring" on page 266 for instructions on configuring the Editor to do this for you.

Navigating the Text

The CodeWarrior editor provides several methods for navigating a file that you are editing.

This section covers these methods:

- Finding a Routine
- Adding, Removing, and Selecting a Marker
- Opening a Related File
- Going to a Particular Line
- Using Go Back and Go Forward
- Configuring Editor Commands
- Opening a File's Directory (Mac OS)

In addition, the integrated code browser has many powerful techniques for navigating through your code. To learn more about using the CodeWarrior Browser, refer to "Browser Overview" on page 201.

You can change the key bindings that cause the text insertion point to move around in a file. Refer to "Editor bindings" on page 255 for more information.

Finding a Routine

Click the Routine icon to display the Routine pop-up menu, discussed in "Routine Pop-Up Menu" on page 132, then select the routine you want to access.

NOTE: If the pop-up is empty, the file is not a source code file.

Adding, Removing, and Selecting a Marker

You can add or remove a marker in any of your text files using the facilities built into the CodeWarrior editor. Markers are like bookmarks. They are useful for setting places in your file that you can jump to quickly, or for leaving notes to yourself about work in progress on your code.

Adding a marker

To add a marker, move the text insertion point to the location in the text you want to mark, then choose **Add marker** from the Marker Pop-Up Menu. A dialog box named Add Marker appears, shown in Figure 5.9.

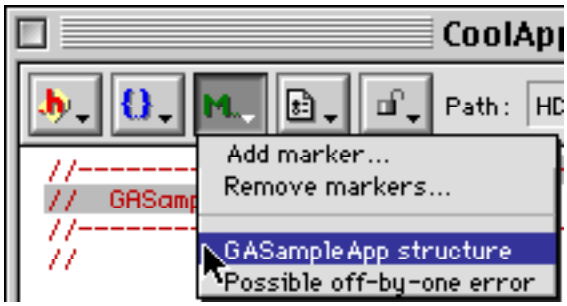
Figure 5.9 Add Marker dialog box

Enter text in the Add Marker dialog box to mark your insertion point location in the file with a note, comment, routine name, or other text that would be helpful to you.

When you are through adding your text note, click **Add** and your marker will be visible in the Marker Pop-Up Menu, as shown in Figure 5.10.

TIP: If you select some text in a source file, then choose Add Marker, the selected text will appear as the new marker name in the Add Marker dialog. This is handy for quickly adding specific routines or lines as markers.

Figure 5.10 Example text File with a marker added



Adding Markers with `#pragma`

There is another method for marking files on a more permanent basis. For C/C++ language programs, use

```
#pragma mark myMarker
```

to leave markers in a file. For Pascal, use

```
{$PRAGMAC MARK myMarker}
```

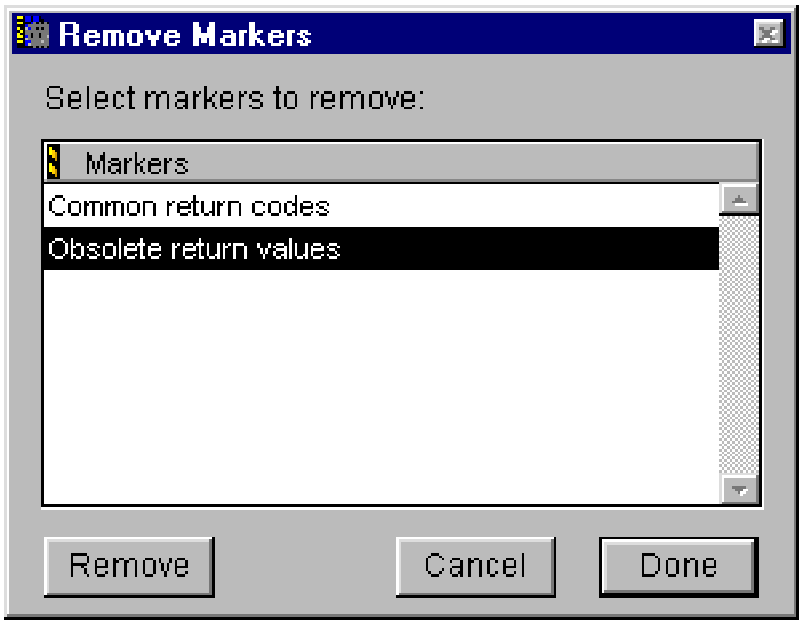
Unlike the markers we've been talking about in this section, these markers don't appear in the Marker Pop-up Menu. Instead, markers created with `#pragma mark` appear in the Routine Pop-up menu.

When embedded in your file, this example adds *myMarker* to the Routine Pop-Up Menu automatically when the file is opened in the Editor.

Removing a marker

To remove a marker, click the Marker Pop-Up Menu and choose the Remove markers command. The dialog box shown in Figure 5.11 is shown, and you may select the marker you wish to delete. After you select the marker, click **Remove** to remove it permanently from the marker list. When finished, click **Done** to close the Remove Markers dialog box.

Figure 5.11 Remove Markers dialog box



Jumping to a marker

Click the Marker Pop-Up Menu and choose the name of the marker from the list shown on the pop-up to set the text insertion point at the location of the marker.

Opening a Related File

There are a few ways to open files related to the active editor window. For example, if you are looking at a C++ `.cpp` source code file and want to view a `.h` header file that is used by the `.cpp` file, there are different ways to do this.

Use the Interface Pop-Up Menu shown in Figure 5.2 on page 132 to open interface or header files referenced by the current file. You can also use the **Touch** and **Untouch** commands from this pop-up.

To open a file in the list, choose the corresponding item from the menu.

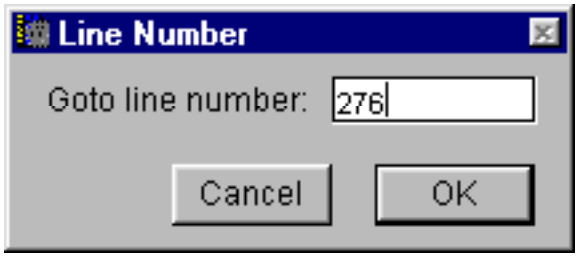
There is another method for opening an interface or header file that your source code file uses. To open the related file, type Control-D (Windows) or Command-D (Mac OS and Solaris) after selecting the file name in the active window. To learn more about this method for opening files, refer to "Opening an Existing File" on page 102.

Going to a Particular Line

You can go to a specific line in an editor window if you know its number. Lines are numbered consecutively, with the first line designated as line 1.

Click the Line Number Button on the editor window to open the Line Number dialog box shown in Figure 5.12. Then enter the number of the line you want to go to and select OK.

Figure 5.12 Line Number dialog box



Using Go Back and Go Forward

The **Go Back** and **Go Forward** commands are only available when you use the Browser. If you already have the Browser enabled, refer to "Go Back and Go Forward" on page 227 for information about how to use these commands.

If you aren't using the Browser and want to learn how to use it, see "Browser Overview" on page 201.

Configuring Editor Commands

The CodeWarrior IDE allows you to customize key bindings for the editor to suit your working style. Refer to "Editor bindings" on page 255 to learn about the commands you can customize to your liking.

Opening a File's Directory (Mac OS)

If you want to know the full directory path for an Editor window file, the Path Pop-Up Menu will show it to you. Press the Command key and click on the file name in the window's title bar to view the Path Pop-up Menu.

You can open the directory that contains the file on display in the active Editor window. Just click the Path Pop-Up Menu (Mac OS) and choose any directory you want to open from the pop-up, as shown in Figure 5.6 on page 137.

Online References

As you're working on a program, you often need to look up the documentation or definition of a particular routine, variable, or type. You may also need to look up documentation about using the IDE. CodeWarrior has both kinds of documentation readily available.

Before you can use the online help available from the CodeWarrior IDE, you need to install the help files and viewer applications from the CodeWarrior Reference CD:

Windows Insert the CodeWarrior Reference CD into your CD-ROM drive and wait for a dialog box to appear. Click "Launch CodeWarrior Setup" and follow the onscreen instructions.

Mac OS Insert the CodeWarrior Reference CD into your CD-ROM drive. Double-click the CodeWarrior Reference Installer icon at the root level of the CD. Follow the onscreen instructions to install the help files and viewer applications.

This section describes how to get started looking up documentation online and how to set up the online reference databases.

The topics in this section are:

- Finding Symbol Definitions
- WinHelp (Windows)
- QuickHelp (Mac OS)
- QuickView (Mac OS)
- THINK Reference (Mac OS)
- Inserting Routine (Reference) Templates (Mac OS)
- HyperHelp (Solaris)

Finding Symbol Definitions

This method lets you find the definition of a symbol in your project source files. If the symbol isn't defined in your project, CodeWarrior finds the symbol's definition in the online documentation viewer selected in IDE Extras preferences panel. Supported online reference viewers include WinHelp (Windows), QuickHelp (Mac OS), QuickView (Mac OS), or THINK Reference (Mac OS).

To look up the definition of a symbol, first select the symbol name in your source code. Then, choose **Find Definition** from the Search Menu. If you don't want to use the Search menu you can instead Alt/Command double-click on the symbol's name. CodeWarrior searches all the files in your project for the definition of the symbol.

If CodeWarrior finds one or more matches in your project, it opens a window and displays each of the matches for you to examine. If the Browser is enabled, the window shown will be a browser Symbol window (see "Symbol Window" on page 222). Otherwise, the window will be a Message window (see "Using the Message Window" on page 340).

Mac OS and Solaris You can also use the **Find Reference** and **Find Definition & Reference** commands to look up symbol definitions. When you choose the Find Reference (Mac OS) command, CodeWarrior searches the on-line documentation selected in IDE

Extras preferences panel (see "IDE Extras Panel" on page 242) for the definition of the selected symbol. If a match is found in the on-line documentation, a viewer is launched and the information displayed. With the Find Definition & Reference (Mac OS) command, CodeWarrior searches the source code files as well as the on-line reference for the definition of the symbol.

TIP: To return to your original location after viewing a symbol's definition, press Shift-Ctrl-B (Windows) or Shift-Command-' (Mac OS).

TIP: (Mac OS and Solaris) Option double-click or hold down the Option key and choose Find Definition to search only your project's files. This prevents CodeWarrior from searching external reference applications for the definition of the symbol.

WinHelp (Windows)

The CodeWarrior IDE provides help files for use with the WinHelp viewer application. WinHelp is part of the Windows operating system.

You can use WinHelp to look up documentation on Win32, MFC, the ANSI libraries, and the CodeWarrior IDE. In addition, you can right-click on many components in the IDE to view pop-up help information.

To use WinHelp, choose **CodeWarrior IDE** from the Help menu. The help file for the IDE is displayed.

To learn about the IDE, simply click the hypertext links in the WinHelp documentation. For information about performing tasks, click the **How do I** button in the WinHelp window. Clicking the **Glossary** button provides descriptions of the terms used in the documentation.

Refer to "Finding Symbol Definitions" on page 156 to begin looking up documentation online.

QuickHelp (Mac OS)

QuickHelp comes with the CodeWarrior IDE. The `CodeWarrior Help` folder contains the help files that are used with QuickHelp.

You can use QuickHelp to look up documentation on PowerPlant, the ANSI libraries, and the CodeWarrior IDE. In addition, you can use Balloon Help on many components in the IDE to view help information.

To use QuickHelp, choose **CodeWarrior IDE** from the Help menu. The help file for the IDE is displayed.

To learn about the IDE, simply click the hypertext links in the QuickHelp documentation. For information about performing tasks, click the **How do I** button in the QuickHelp window. Clicking the **Glossary** button provides descriptions of the terms used in the documentation.

Refer to "Finding Symbol Definitions" on page 156 to begin looking up documentation online.

QuickView (Mac OS)

QuickView is part of the *Macintosh Programmer's Toolbox Assistant* (MPTA) published by Addison-Wesley for Apple Computer.

You can use QuickView to look up documentation on PowerPlant, the ANSI libraries, and the CodeWarrior IDE.

If you have the *Macintosh Programmer's Toolbox Assistant*, you can also look up documentation on Mac OS Toolbox routines.

To use QuickView with the CodeWarrior IDE, it is necessary to first set up all the QuickView files properly:

1. **Position your files.**

   Place all your QuickView files in the same folder as the QuickView application that comes with the CodeWarrior IDE. If you already have the Macintosh Programmer's Toolbox Assistant (MPTA), copy all its QuickView files into the same folder that your CodeWarrior QuickView files are in.

2. **Remove multiple copies of the QuickView application.**

   Keep the QuickView application that came with the CodeWarrior IDE. Delete all other QuickView applications on your computer system.

3. **Select QuickView as the on-line reference.**

   Refer to "IDE Extras Panel" on page 242 to learn how to select the QuickView application as the online reference database for your project.

   Refer to "Inserting Routine (Reference) Templates (Mac OS)" on page 160 or "Finding Symbol Definitions" on page 156 to begin looking up documentation online.

THINK Reference (Mac OS)

To use THINK Reference with the CodeWarrior IDE, refer to "IDE Extras Panel" on page 242 to enable the THINK Reference application as the online reference database for your project.

You must have the THINK Reference database product properly installed on your computer system. THINK Reference is not part of the CodeWarrior product. THINK Reference is available from Developer Depot on the MacTech CD-ROM and includes several databases on Mac OS Toolbox routines.

Refer to "Inserting Routine (Reference) Templates (Mac OS)" on page 160 or "Finding Symbol Definitions" on page 156 to begin looking up documentation online.

Inserting Routine (Reference) Templates (Mac OS)

If you're looking up a routine (such as an operating system call) in the online reference database, you can paste the template for the call into your Editor window at the text insertion point. This is useful when you know the name of the call you want to add to your source code, but you don't know what its parameters are supposed to be.

A routine template looks like this:

```
SetRect(r, left, top, right, bottom);
```

To insert a reference template into your code, type the routine name that you want to insert, then select the name you just typed. Finally, choose Insert Reference Template (Mac OS) from the Edit Menu.

The CodeWarrior IDE searches for the routine in either QuickView (Mac OS) or THINK Reference (Mac OS), starting the required application if it isn't already running. If the routine is found, the template is copied into your Editor window and replaces the text you selected with the template. If the definition is not found, you will hear a beep.

HyperHelp (Solaris)

HyperHelp support will be available in future versions of the CodeWarrior IDE for Solaris.

6

Searching and Replacing Text

This chapter explains how to use the CodeWarrior IDE facilities to search and replace text in files.

Searching and Replacing Text Overview

The CodeWarrior IDE provides comprehensive search and replace features with the Find dialog box. You can search for and replace text in a single file, in every file in a project, or in any combination of files. You can also search for regular expressions, like as those used in UNIX's `grep` command.

The topics in this chapter are:

- Guided Tour of the Find Dialog Box
- Finding and Replacing Text in a Single File
- Finding and Replacing Text in Multiple Files
- Searching for Selected Text
- Using Regular Expressions (grep)

Guided Tour of the Find Dialog Box

The Find window, shown in Figure 6.1 on page 163 (Windows) and Figure 6.4 on page 169 (Mac OS and Solaris), is a versatile feature of the CodeWarrior IDE. To show the Find dialog box, choose the **Find** command in the Search Menu. Use this dialog box to perform find and replace operations for text in a single file or for text in multiple files in your project. You can use text strings, text substrings, and pattern matching to carry out find and replace operations.

There are two different sections in the Find dialog box.

- Find and Replace Section
- Multi-File Search Section

Find and Replace Section

This section presents a short tour of the find and replace user interface items in the Find window shown in Figure 6.1 on page 163. The items in the window are:

- Find text box
- Replace text box
- Recent Strings pop-up menu
- Find button
- Replace button
- Replace & Find button
- Replace All button
- Batch checkbox
- Wrap checkbox
- Ignore Case checkbox
- Entire Word checkbox
- Regexp checkbox
- Multi-File Search Disclosure triangle
- Multi-File Search button

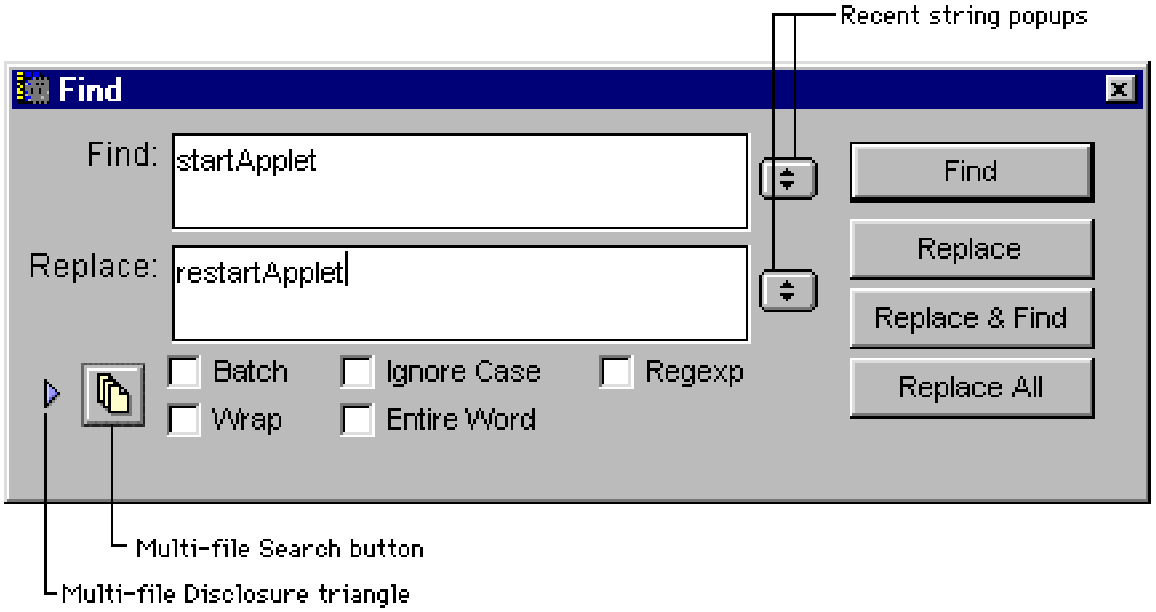
Find text box

The Find text box is one of the editable text fields in the Find dialog box, shown in Figure 6.1. You enter text in this field that you want to search for.

You can use the Cut, Copy, Paste, and Clear commands with the Find text box. These commands are documented in the section called "Edit Menu" on page 379.

Also, the discussion "Enter 'Find' String" on page 384 tells how to enter text into the Find Text Box without using the Find dialog box.

Figure 6.1 The Find Dialog search and replace section



Replace text box

The Replace text box is one of the editable text fields in the Find dialog box, shown in Figure 6.1. The text you enter in this field will be used to replace the text you're searching for.

You can use the Cut, Copy, Paste, and Clear commands with the Find Text Box. These commands are documented in the section called "Edit Menu" on page 379.

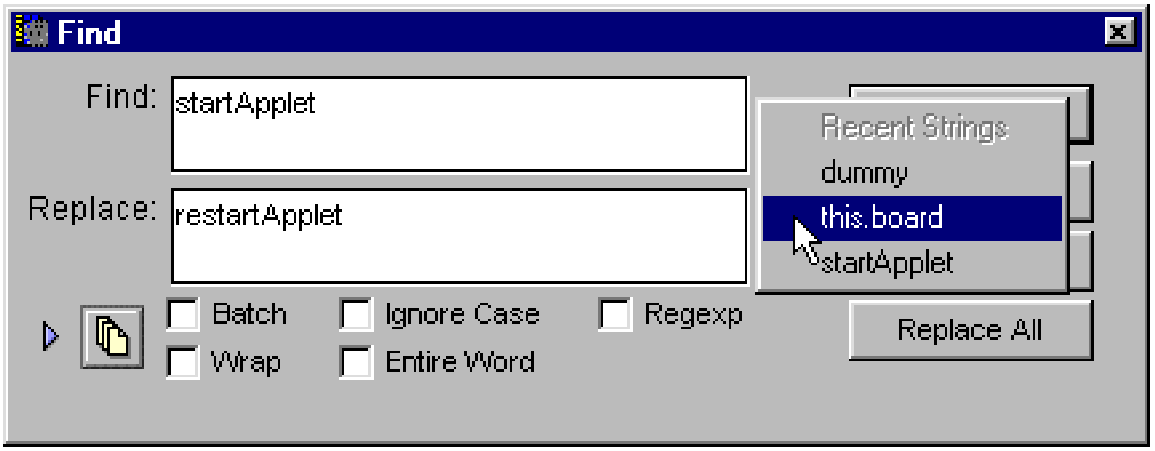
Also, the discussion "Enter 'Replace' String" on page 384 tells how to enter text into the Replace Text Box without using the Find dialog box.

Recent Strings pop-up menu

The Recent Strings pop-up menu is shown in Figure 6.2. It contains strings that were recently used for searches.

There are actually two of these pop-ups. Each pop-up is to the right of both the Find text box and the Replace text box. Selecting an item in one of these pop-ups enters it in the corresponding text box.

Figure 6.2 Recent Strings pop-up menu



Find button

The **Find** button, shown in Figure 6.1 on page 163, allows you to begin a text search operation once you set the other Find dialog box controls, and have completed certain required fields in the Find dialog box.

To learn more about finding text, see "Searching for Selected Text" on page 172.

Replace button

The **Replace** button is one of the buttons in the Find dialog box, shown in Figure 6.1 on page 163.

When you enter text in the Find text box and click the **Find** button, the CodeWarrior IDE will search for matching text according to the control settings checked at the bottom of the Find dialog box. If a match is found for the text in the Find text box, the **Replace** button can be clicked to replace the found text with that shown in the Replace text box.

To learn more about searching and replacing text, see "Replacing Found Text" on page 178.

Replace & Find button

The **Replace & Find** button is shown in Figure 6.1 on page 163. This button behaves much like the Replace button, but also initiates another Find operation after the text substitution is performed.

To learn more about searching and replacing text, see "Finding and Replacing Text in a Single File" on page 174 and "Finding and Replacing Text in Multiple Files" on page 181.

Replace All button

The **Replace All** button is shown in Figure 6.1 on page 163. This button behaves much like the Replace button, but replaces *every* occurrence of the text shown in the Find text box with the text shown in the Replace text box.

To learn more about searching and replacing text, see "Replacing Found Text" on page 178.

Batch checkbox

The **Batch** checkbox is shown in Figure 6.1 on page 163. Selecting this checkbox causes the results of the **Find** command to appear in a Search Results message window (Figure 6.7 on page 181).

To learn more about the role of the Batch checkbox in searching, see "Using Batch Searches" on page 180.

Wrap checkbox

The **Wrap** checkbox is shown in Figure 6.1 on page 163. Selecting this checkbox allows a search that reaches one end of a file or group of files to continue from the opposite end.

To learn more about this feature, consult "Controlling Search Range in a Single File" on page 176.

Ignore Case checkbox

The **Ignore Case** checkbox is shown in Figure 6.1 on page 163. This checkbox causes the CodeWarrior IDE to disregard the case (uppercase or lowercase) of the text entered into the Find text box.

To learn more about this feature, consult "Controlling Search Parameters" on page 177.

Entire Word checkbox

The **Entire Word** checkbox is shown in Figure 6.1 on page 163. This checkbox causes the CodeWarrior IDE to ignore matching text that occurs within words.

To learn more about this feature, consult "Controlling Search Parameters" on page 177.

Regexp checkbox

The **Regexp** checkbox is shown in Figure 6.1 on page 163. This checkbox causes the CodeWarrior IDE to interpret the text in the Find text box as a regular expression.

CodeWarrior's regular expressions are similar to the regular expression for `grep` in UNIX™. To learn more about this feature, refer to "Using Regular Expressions (grep)" on page 191.

Multi-File Search Disclosure triangle

The Multi-File Search Disclosure triangle is shown in Figure 6.1 on page 163. Click this triangle to show the Multi-File Search Section of the Find window, as shown in Figure 6.3 on page 167 (Windows) or Figure 6.4 on page 169 (Mac OS and Solaris).

To learn more about multi-file searching with the Find window, see "Finding and Replacing Text in Multiple Files" on page 181.

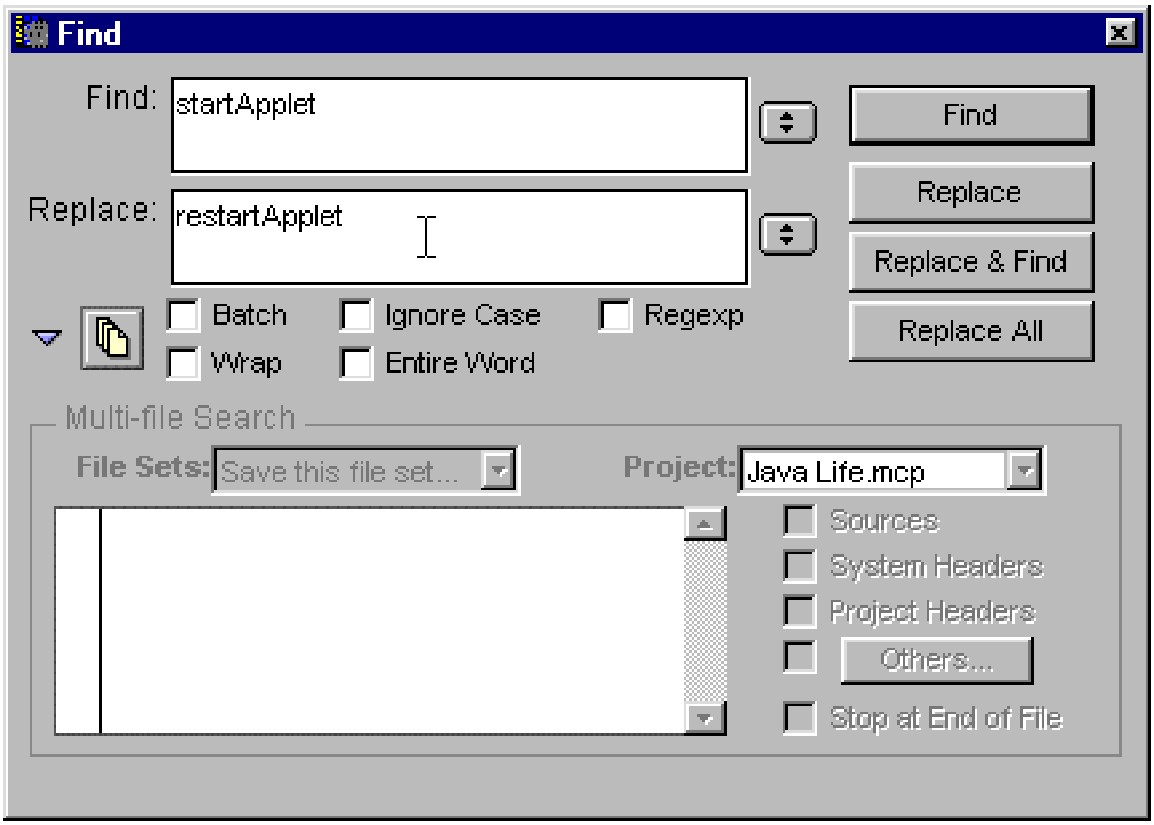
Multi-File Search button

The Multi-File Search button is shown in Figure 6.3. Click this button to enable or disable the options in the Multi-File Search Section of the Find window, shown in Figure 6.4 on page 169.

When the Multi-File Search button is not depressed, as shown in the dialog box of Figure 6.3, the items in the Multi-File Search Section of the Find window are dimmed.

To learn more about the Multi-File Search Button, see "Activating Multi-File Search" on page 182.

Figure 6.3 Multi-File Search button not selected



Multi-File Search Section

This section describes the user interface items in the Multi-File Search Section of the Find window, shown in Figure 6.4. The user interface items are:

- File list
- File Sets pop-up menu
- Project pop-up menu
- Stop at End of File checkbox
- Sources checkbox
- System Headers checkbox
- Project Headers checkbox
- Others button

File list

The File list is shown in Figure 6.4. This is a list of the files that will be searched in a Multi-file search. You add files to this list by enabling the **Sources**, **System Headers**, **Project Headers**, and **Others** controls. You can also drag and drop groups or files from the Project window into the list.

For more information about adding files and removing files in file sets, see "Choosing Files to be Searched" on page 183.

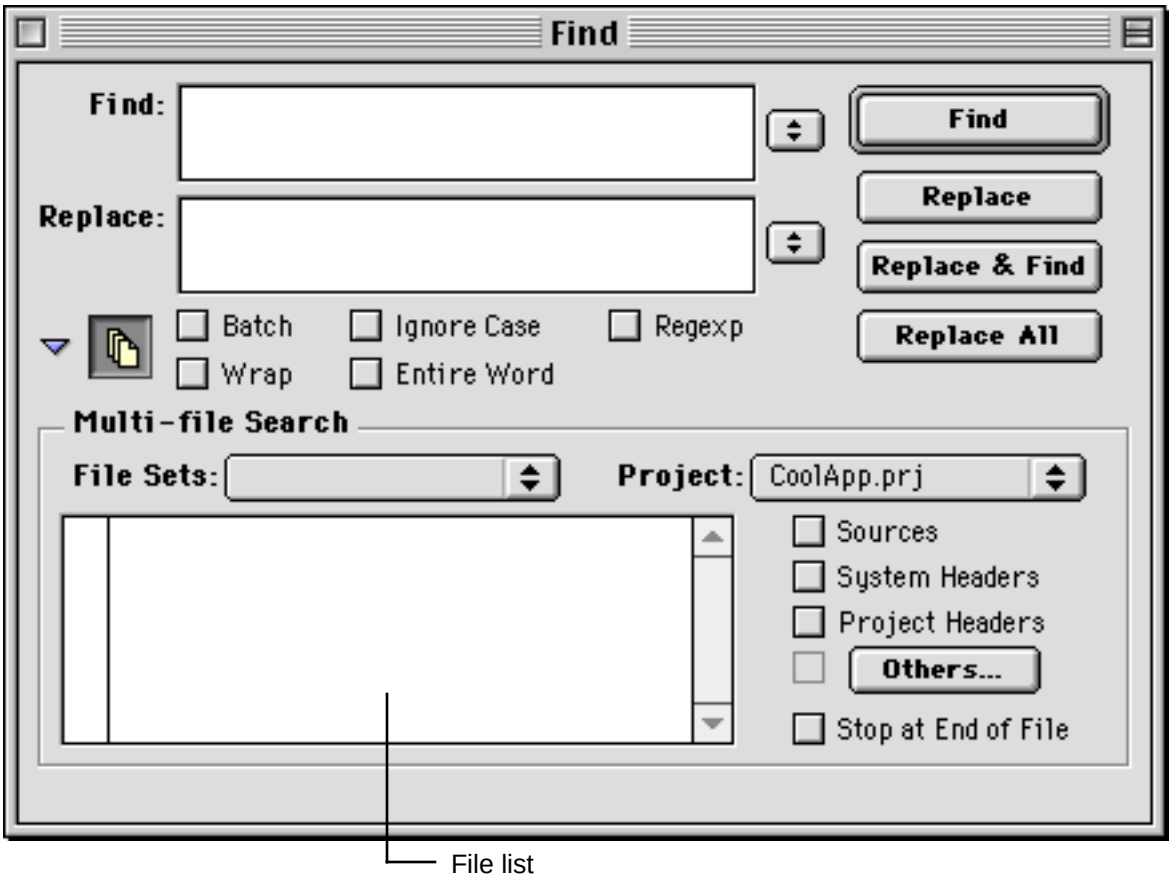
File Sets pop-up menu

The File Sets pop-up menu is shown in Figure 6.5. This pop-up menu is used with Multi-file searches. Use this pop-up menu to select, add, and remove saved sets of files to search and replace.

You can build up sets of files, such as collections of header or interface files, that will be available whenever you want to search through the files for text.

For more information about Multi-file sets, see "Choosing Files to be Searched" on page 183.

Figure 6.4 The Find Dialog Box for a multiple file search



Project pop-up menu

The Project pop-up menu, shown in Figure 6.6, allows you to choose the project file with which you want to perform your search. Since the CodeWarrior IDE can have multiple projects open at a time, this menu provides a way to perform the same search in different projects.

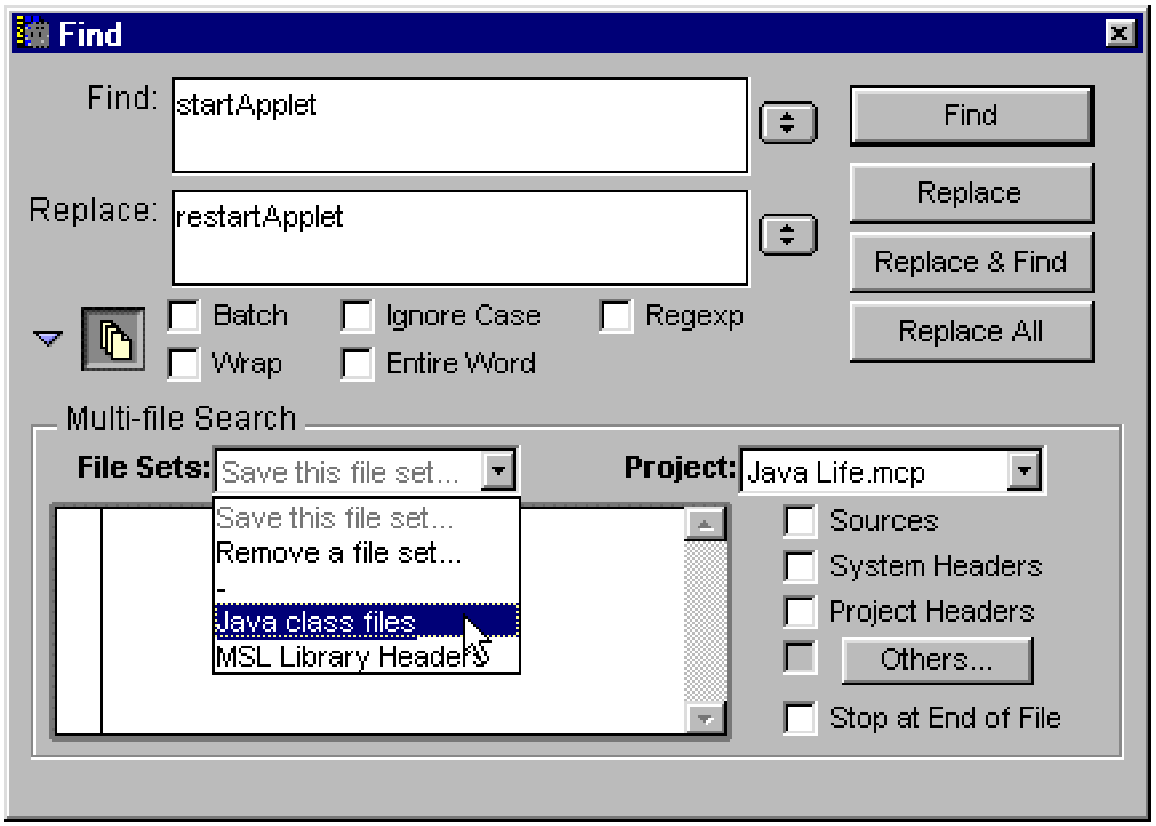
Stop at End of File checkbox

If you turn off the **Stop at End of File** checkbox, all the files in the File list are searched as one large file. When the CodeWarrior IDE reaches the end of one file, it starts searching the next. When it reaches the end of the last file to search, it beeps.

To search each file individually, enable the **Stop at End of File** checkbox. When the CodeWarrior IDE reaches the end of a file, it stops searching and beeps. You must choose **Find in Next File** from the Search Menu to continue the search.

For more information about using the Stop at End of File checkbox, see "Controlling Search Range in a Single File" on page 176.

Figure 6.5 File Sets pop-up menu

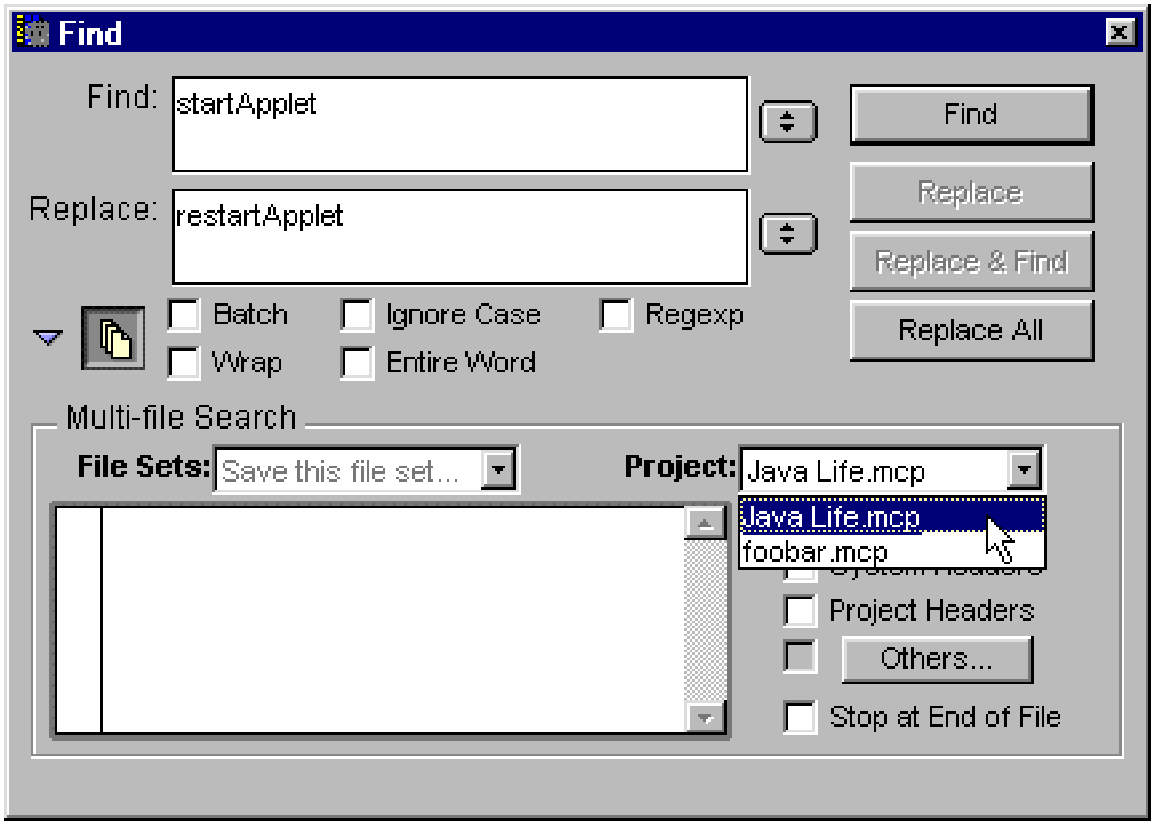


Sources checkbox

The **Sources** checkbox is shown in Figure 6.4 on page 169. This checkbox adds all the source files from the current project to the File list.

For more information about source files in file sets, and their role in Multi-file searches, see "Adding project source files" on page 183.

Figure 6.6 Project Pop-up Menu



System Headers checkbox

The **System Headers** checkbox is shown in Figure 6.4 on page 169. This checkbox adds all system header or interface files from the current project to the File list.

For more information about system headers in file sets, and their role in multi-file searches, see "Adding system header files" on page 184.

Project Headers checkbox

The **Project Headers** checkbox is shown in Figure 6.4 on page 169. This checkbox adds all the project header or interface files from the current project to the File list.

For more information about project headers in file sets, and their role in Multi-file searches, see "Adding project header files" on page 184.

Others button

The **Others** button is shown in Figure 6.4 on page 169. This button and its checkbox allows you to add one or many additional files to the File list.

For more information about adding file to file sets, see "Adding and removing arbitrary files" on page 184.

Searching for Selected Text

The IDE provides two ways of searching for text without using the Find dialog box. In both of these methods, you select text in a window, and the CodeWarrior IDE finds the text for you without displaying the Find dialog box.

When you search for selected text, the CodeWarrior IDE uses the option settings that you last chose in the Find window. To change these option settings, you must use the Find dialog box.

You should know how to select text in the editor window before reading this section. If you don't know how to select text, refer to "Selecting Text" on page 144.

Finding text in the active editor window

This method is useful if you want to find additional occurrences of a text string in the same open editor window that you're working with.

First, select an instance of the text you want to find. After selecting your text, choose **Find Selection** from the Search Menu.

The CodeWarrior IDE looks for the next occurrence of your text string in the current file only.

To search toward the end of the file for the next occurrence of the text string, click the **Find** button or choose **Find Next** from the Search Menu. You can also press the keyboard shortcut shown to the right of the Find Next command in the Search menu.

To search toward the beginning of the file for the previous occurrence of the text string, use the **Find Previous** command:

Windows This command is available by pressing the key binding Shift-F3.

Mac OS and Solaris This command is available when you hold down the Shift key while using the Search Menu.

The CodeWarrior IDE finds the previous occurrence of the text string and selects it. If the string is not found, the CodeWarrior IDE beeps.

Search for more occurrences of the text string by continuing to use Find, Find Next, or Find Previous on the Search Menu.

Finding text in another window

This method is useful when your text string is in one file and you want to search for the same text string in another file.

First, select an instance of the text you want to find. After selecting your text, choose **Enter 'Find' String** from the Search Menu. The editor enters the text in the Find text box of the Find dialog box.

Now make the window you want to search active, Then, choose **Find Next** from the Search Menu to search forwards in the active editor window for the next occurrence of your text string.

Use the **Find Previous** command to search backwards in the active editor window for the next occurrence of your text string:

Windows This command is available by pressing the key binding Shift-F3.

Mac OS and Solaris This command is available when you hold down the Shift key while using the Search Menu.

The CodeWarrior IDE looks for the Find text box string in the active editor window, starting from the location of the text insertion point in that window.

If you want to search toward the end of the file for the next occurrence of the Find text box string, click the **Find** button or choose **Find Next** from the Search Menu. You can also press the keyboard shortcut shown to the right of the Find and Find Next commands in the Search menu.

To search toward the beginning of the file for the previous occurrence of the Find string, use the **Find Previous** command.

Search for more occurrences of the Find text box string by continuing to use Find, Find Next, or Find Previous from the Search Menu.

Finding and Replacing Text in a Single File

The Find window allows you to search for text patterns in the active editor window. When you find the text you are interested in, you can change it or look for another occurrence of it.

This section discusses how to use the Find dialog box to locate specific text you want to replace in the active editor window.

If you don't yet have a window open, see "Opening an Existing File" on page 102.

If you haven't yet created a file, see "Creating a New File" on page 101.

The topics in this section are:

- Finding Search Text
- Controlling Search Range in a Single File

- Controlling Search Parameters
- Searching with Special Characters
- Replacing Found Text
- Using Batch Searches

Finding Search Text

To enter text in the Find text box, bring up the Find window using the **Find** command in the Search menu. Type a text string into the Find text box on the dialog box, or choose a string from the Recent Strings pop-up menu, as shown in Figure 6.2 on page 164.

NOTE: To learn how to enter search text containing a Return or Tab character, refer to "Searching with Special Characters" on page 178.

Before searching, you can set other search options that control the range of your search.

The search range defines whether you want to search the entire file or just from the text insertion point in one direction. To set up the range of your search, see "Controlling Search Range in a Single File" on page 176.

The search parameters define whether you want to search for text regardless of upper or lower case, and whether to search partial words for the text. To set up the parameters of your search, see "Controlling Search Parameters" on page 177.

Before proceeding, make sure that multi-file searching is turned off since you are only interested in searching the active editor window. To learn about how to determine whether multi-file searching is turned off, refer to "Activating Multi-File Search" on page 182.

Click the **Find** button in the Find window to search forward from the text insertion point in the file, or choose **Find** or **Find Next** from the Search Menu. CodeWarrior now searches for the Find text box string in the active editor window.

Windows Press the key binding Shift-F3 to perform a **Find Previous** command, which searches backwards from the text insertion point in the file.

Mac OS and Solaris Hold down the Shift key and choose **Find Previous** from the Search Menu if you want to search backwards from the text insertion point in the file.

To continue searching toward the end of the file for the next occurrence of the Find text box string, click the **Find** button or choose **Find Next** from the Search menu.

To continue searching toward the beginning of the file for the previous occurrence of the Find text box string, use the **Find Previous** command.

The editor finds and selects the Find text box string. If the string is not found, the editor beeps.

Search for more occurrences of the Find text box string by continuing to use Find, Find Next, or Find Previous.

From this point, you can replace some or all of the text you find with a new text string.

To replace text, see "Replacing Found Text" on page 178.

Controlling Search Range in a Single File

The **Wrap** checkbox option in the Find dialog box controls the flow of the search when you reach the beginning or end of a file.

For example, suppose that you enable the Wrap checkbox option. When the text insertion point is somewhere in the middle of a source file in the active editor window, and you choose **Find Next** on the Search Menu, the CodeWarrior IDE searches from the insertion point to the end of the file. When the search reaches the end of the file, the search continues from the beginning of the file to the insertion point. In other words, the search "wraps" around the ends of the file.

The **Find Previous** command operates in a similar fashion. When the search reaches the beginning of the file and the Wrap checkbox option is enabled, the search continues from the end of the file.

If you have the Wrap checkbox option unchecked, and you choose Find Next on the Search Menu, the search stops when it reaches the end of the file.

If you're searching multiple files with the Wrap checkbox option checked, the CodeWarrior IDE searches from the first file in the file list after it reaches the last file.

Controlling Search Parameters

The **Ignore Case** checkbox and the **Entire Word** checkbox are two easily-accessible options in the CodeWarrior IDE that you can use for matching text.

Ignore Case checkbox

The **Ignore Case** checkbox is shown in Figure 6.1 on page 163. This checkbox causes the CodeWarrior IDE to disregard the case (upper or lower) entered into the Find text box.

For example, if "Foobar" is in the Find text box, then the CodeWarrior IDE will also find occurrences like "foobar" or "FOOBAR", and other possible combinations of upper and lower-case text characters.

Entire Word checkbox

The **Entire Word** checkbox is shown in Figure 6.1 on page 163. This checkbox causes the CodeWarrior IDE to exactly match the text in the Find text box instead of matching similar variations of the text. For example, if the Find text box string is "Word", the CodeWarrior IDE finds only "Word" in the file. If this option is off, the same text matches with variations like "Words", "WordCount", and "BigWordCount".

Searching with Special Characters

To enter a Tab or Return character in the Find or Replace fields, use one of the following methods:

- Copy and paste your selected text with the Tab or Return characters into the Find or Replace field, or
- If drag and drop is supported, drag the text from the Editor window directly into a field, or
- Enable the **Regexp** option and enter `\t` for Tab or `\r` for Return into the field.

Mac OS and Solaris To directly enter a Tab character, press Option-Tab. To enter a Return character, press Option-Return.

WARNING! Using **Regexp** will alter the manner in which CodeWarrior locates a string match. See "Using Regular Expressions (grep)" on page 191 for more information on using Regexp.

Replacing Found Text

When you find an occurrence of text you are interested in, you can either replace one occurrence at a time, or you can replace all occurrences in the entire file.

Selective Replace

To selectively replace text, first enter some text to find, then choose the **Find** operation on the Search Menu, or click the Find button in the Find dialog box. You can read more about how to find text by referring to "Finding Search Text" on page 175.

Next, enter the replacement text string in the Replace text box field of the Find dialog box.

Type the string in the Replace text box field or choose a string from the Recent Strings pop-up menu of the Replace text box by clicking the arrow icon just to the right. The Recent Strings pop-up menu (Figure 6.2 on page 164) contains the last five strings you have used.

Now choose whether to replace the string you found. For convenience, there are three buttons in the Find dialog box for doing this, the **Replace** button, the **Replace & Find** button, and the **Replace All** button. Each button performs a different operation.

To replace the string and see the results, click the Replace button in the Find dialog box or choose **Replace** from the Search Menu. The editor replaces the text that was found with the Replace text box string.

To continue searching forward, choose **Find Next** from the Search Menu, or click the Find button in the Find dialog box.

To continue searching backward, use the **Find Previous** command, or press the Shift key and click the Find button in the Find dialog box.

To replace the string and find the *next* occurrence, choose **Replace & Find Next** from the Search Menu, or click the Replace & Find button in the Find dialog box. The Editor replaces the selected text with the Replace text box string and finds the next occurrence of the Find text box string. If the Editor can't find another occurrence, it beeps.

To replace the Find text box string and find the *previous* occurrence, use the Replace & Find Previous command:

Windows Press the key binding Ctrl-Shift-L, or press the Shift key as you click the **Replace & Find button** in the Find dialog box.

Mac OS hold down the Shift key as you choose Replace & Find Previous from the Search Menu, or press the Shift key as you click the **Replace & Find button** in the Find dialog box.

The Editor replaces the selected text with the Find text box string and searches for a previous occurrence of the Find text box string. If the CodeWarrior IDE can't find another occurrence, it beeps.

Replace All

To replace text, first enter some text to find in the Find text box, then choose the **Find** operation on the Search Menu, or click the Find but-

ton in the Find dialog box. You can read more about how to find text by referring to "Finding Search Text" on page 175.

Next, enter the replacement text string in the Replace text box field of the Find dialog box.

To replace all the occurrences of the Find text box string, click the Replace All button in the Find dialog box, or choose **Replace All** from the Search Menu.

WARNING! Be careful when you use the **Replace All** command, since **Undo** is not available for this operation.

TIP: If you are going to perform a **Replace All** operation on a single source file, make sure to save the source file before executing the replace operation. In the event that you should change your mind, and before you save any changes, use **Revert** to replace the modified file in memory with the saved version on disk. This technique will not work across multiple files.

Using Batch Searches

The CodeWarrior IDE gives you a way to collect all matching descriptions of your text search in one window for easy reference.

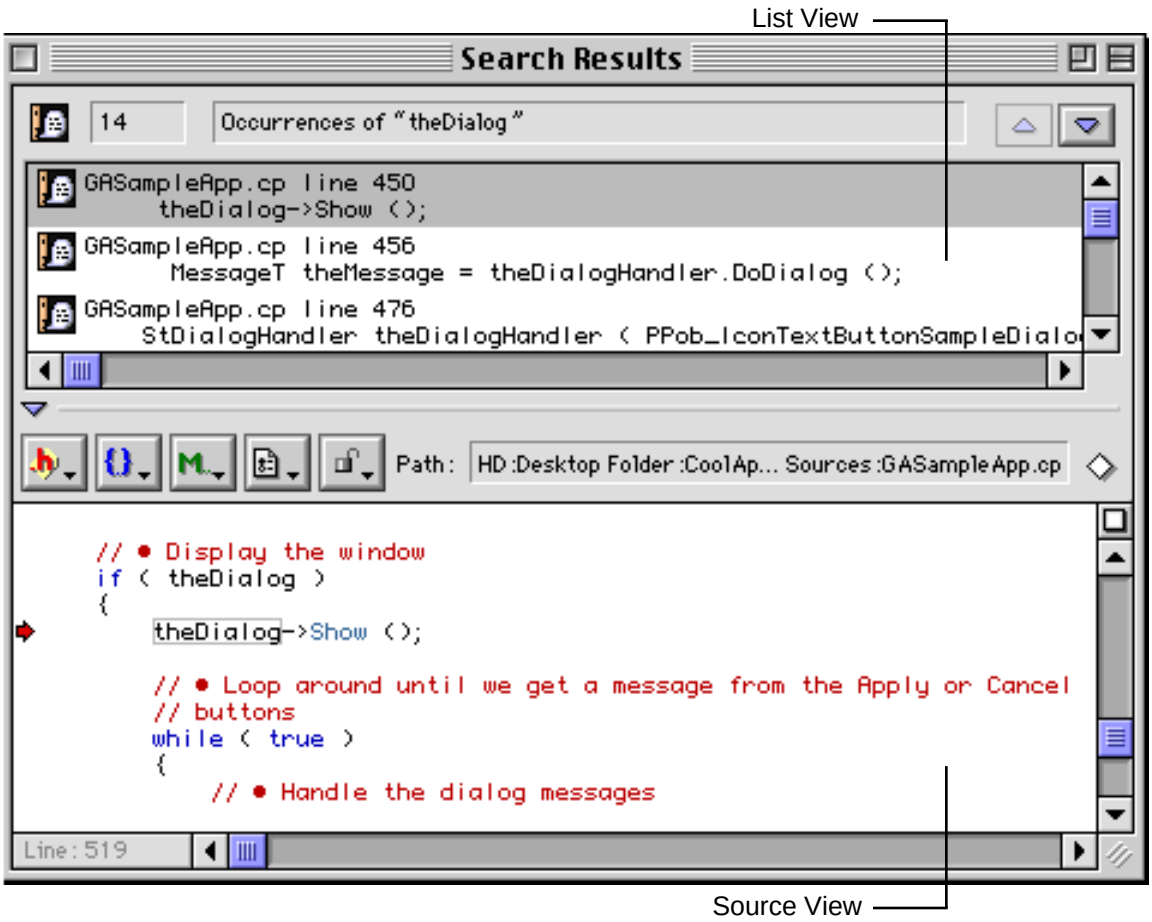
If the Batch checkbox option is checked in the Find dialog box, and the **Find** button is clicked, the CodeWarrior IDE searches for all occurrences of the Find text box string and lists them in the Search Results message window, as shown in Figure 6.7.

The Search Results window shown in Figure 6.7 has a List View and a Source View.

To go to a particular occurrence of the Find text box string, so that it is shown in the Source View pane of the window, double-click on its entry in the List View.

To learn more about the features of this window, refer to the discussion of the Message Window in "Guided Tour of the Message Window" on page 337.

Figure 6.7 Batch search results



Finding and Replacing Text in Multiple Files

The CodeWarrior IDE allows you to search multiple files for the occurrence of text strings.

In this section you will learn how to do text searches through multiple files.

Another way to quickly access information and search in multiple files is with the Browser's **Go Back** and **Go Forward** commands on the Search menu. To learn about how to use these commands, refer to "Go Back and Go Forward" on page 227.

The topics in this section are:

- Activating Multi-File Search
- Choosing Files to be Searched
- Saving a File Set
- Removing a File Set
- Controlling Search Range in Multiple Files

Activating Multi-File Search

To configure the CodeWarrior IDE to search through multiple files, you need to activate multi-file searching in the Find dialog box.

When the Multi-File Search button is on, the button appears to be depressed.

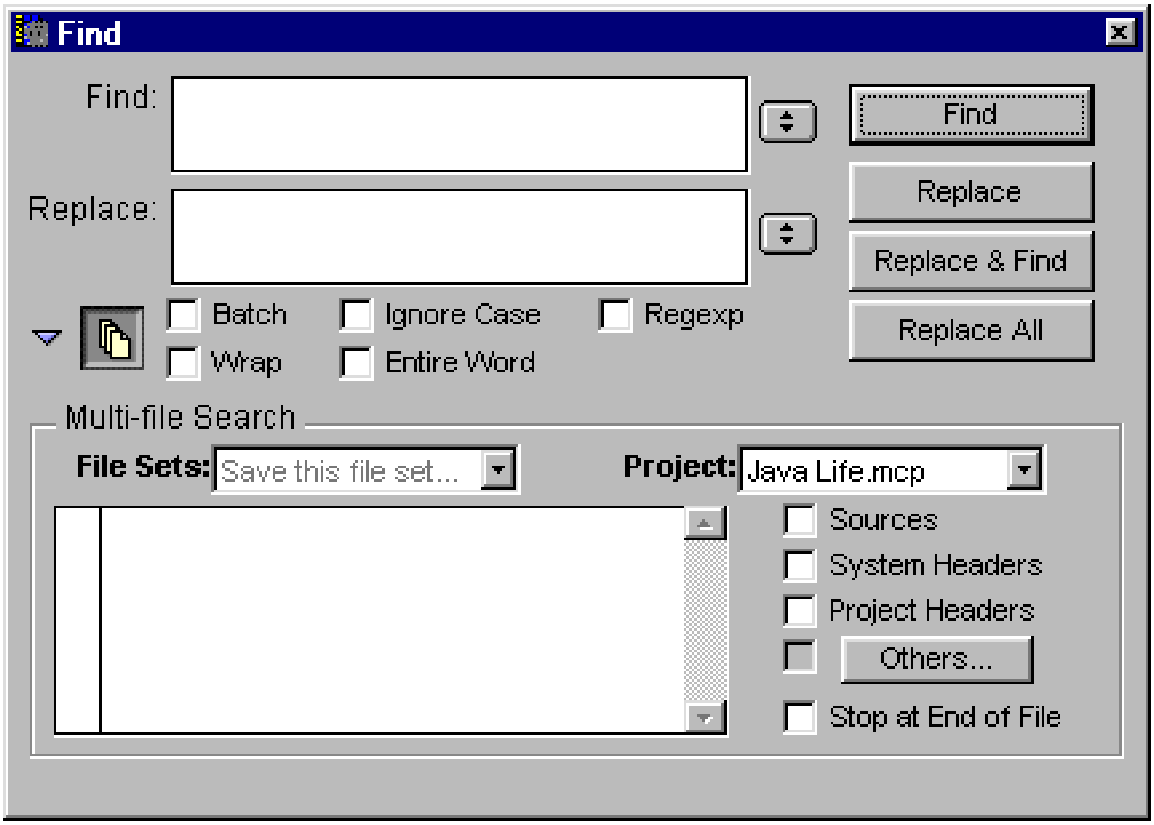
When the Multi-File Search button is off, the button looks three-dimensional.

Click the Multi-File Search Disclosure triangle to the left of the Multi-File Search button, shown in Figure 6.1 on page 163, so that the triangle points down.

The CodeWarrior IDE displays the Multi-File Search Section, an extension of the Find dialog box, as shown in Figure 6.8 on page 183. When the Multi-File Search button is on (as shown above), the Multi-File Search Section is enabled.

To learn about how to configure the Multi-File Search Section of the Find dialog box, refer to "Choosing Files to be Searched" on page 183, "Saving a File Set" on page 187, "Removing a File Set" on page 188, and "Controlling Search Range in Multiple Files" on page 189.

Figure 6.8 The Find dialog box with multi-file search options



Choosing Files to be Searched

There are several ways to choose files for a search.

Adding project source files

To add all the source files from the current project, turn on the **Sources** checkbox. When you turn off the Sources checkbox, the CodeWarrior IDE removes all associated files from the file list.

To include only some of the files, turn on the Sources checkbox and delete the files you don't want by selecting them and pressing Backspace/Delete after clicking on the file name.

If turning on this option doesn't add any files, update your project's internal list of header and interface files with the Make command. To learn how to do this, refer to "Making a Project" on page 323.

Adding project header files

To add all the project header or interface files from the current project, turn on the **Project Headers** checkbox. When you turn off the Project Headers checkbox, the CodeWarrior IDE removes all associated files from the file list.

To include only some of the files, turn on the Project Headers checkbox and delete the files you don't want by selecting them and pressing Backspace/Delete.

If turning on this option doesn't add any files, update your project's internal list of header or interface files with the Make command. To learn how to do this, refer to "Making a Project" on page 323.

Adding system header files

To add all the system header or interface files from the current project, turn on the **System Headers** checkbox. When you turn off the System Headers checkbox, the CodeWarrior IDE removes all associated files from the file list.

To include only some of the files, turn on the System Headers checkbox and delete the files you don't want by selecting them and pressing Delete.

If turning on this option doesn't add any files, update your project's internal list of header or interface files with the Make command. To learn how to do this, refer to "Making a Project" on page 323.

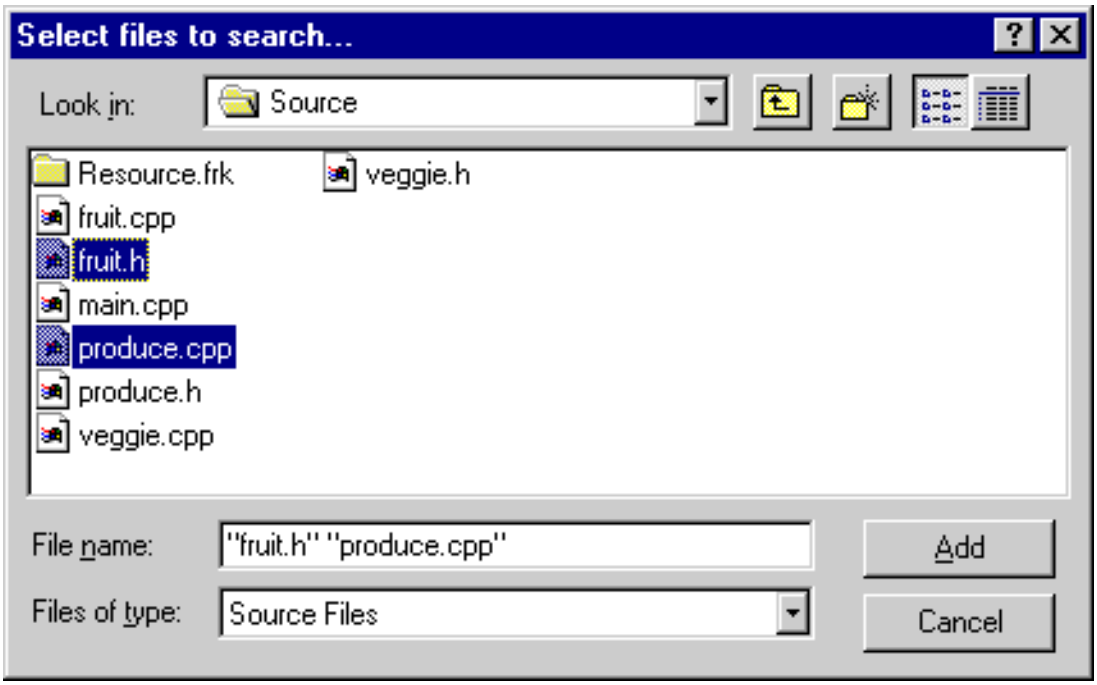
Adding and removing arbitrary files

For multi-file searches, you can manually add and remove files to your file set. Adding files to a file set enables you to search in files not included in your current project.

First, click the Others button in the Multi-File Search Section of the Find dialog box. The Select Files To Search dialog box will be dis-

played as shown in Figure 6.9 (Windows), Figure 6.11 (Mac OS), and Figure 6.12 (Solaris).

Figure 6.9 The Select Files To Search dialog box (Windows)



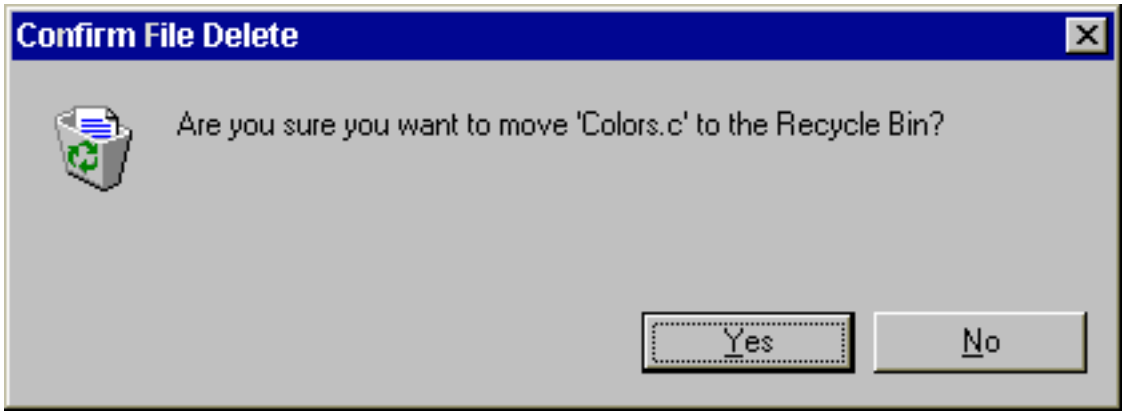
The Select Files To Search dialog box displays a list of the files in the current directory that may be added to the file set. To add a file to the file set, select it and click the **Add** button. If you change your mind, click **Cancel** and the file set will not be changed.

Windows You can select multiple files in this list by pressing the Control key and clicking a file simultaneously. When you're finished choosing files, click **Add**. All files in the Select Files To Search list will appear in the file set. To remove files from the file set, select them in the Find dialog box (Figure 6.8 on page 183) and press Backspace or Delete.

WARNING! (Windows) If you select a file's icon in the Select Files To Search dialog box (Figure 6.9) and press the Delete key, the Confirm File Delete dialog box (Figure 6.10) will appear. This

dialog box warns you that you are attempting to delete the selected file from your hard drive. Click **No** to return to the Select Files To Search dialog box. Instead of selecting the file's icon, select the file's name (including the quote marks) in the File Name text box and press Backspace or Delete. This is the correct way to remove a selected file from the Select Files To Search dialog box.

Figure 6.10 Confirm File Delete dialog box (Windows)



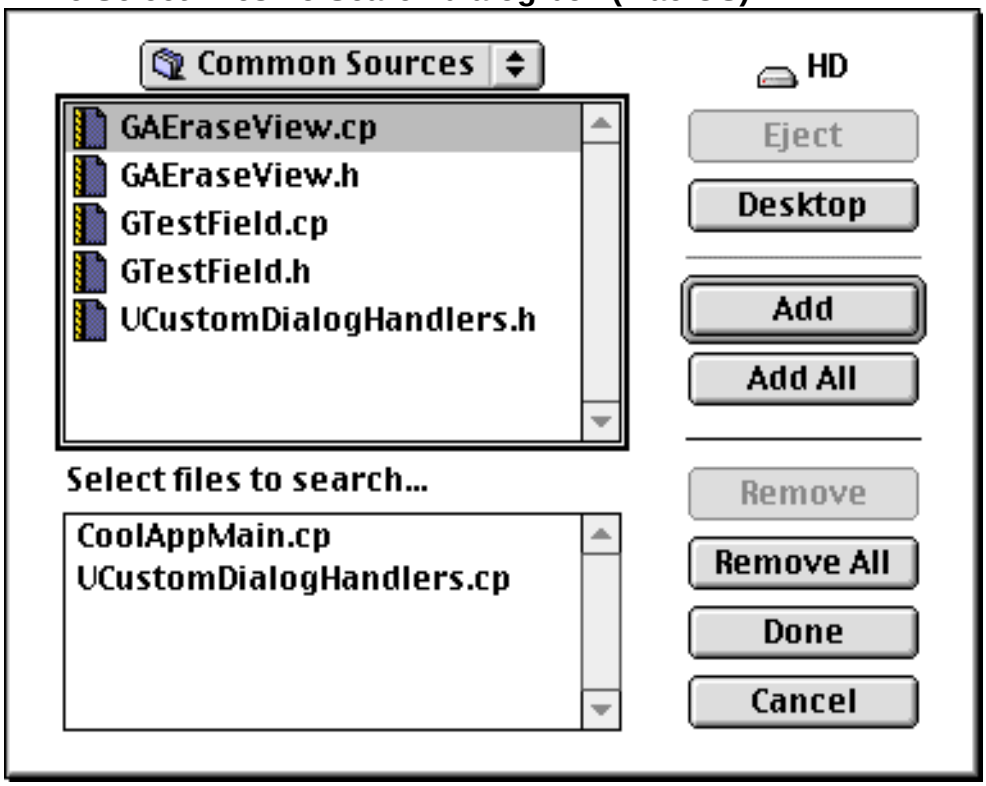
Mac OS and Solaris To add all files in the list, click the **Add All** button. The files are removed from the top file list and reappear in the Select Files To Search list at the bottom of the dialog box. To remove files from the Select Files To Search list, select them and click the **Remove** button. To remove all the files from the Select Files To Search list, click the **Remove All** button. When you're finished choosing files, click **Done**. All files in the Select Files To Search list will appear in the file set.

To add more files later, just click the Others button in the Find dialog box and repeat the process.

Choosing a file set

To select a previously-saved file set to include in your search, click on the File Sets pop-up menu and choose a file set from the menu, as shown in Figure 6.5 on page 170. The files then appear in the File Sets list.

Figure 6.11 The Select Files To Search dialog box (Mac OS)



Saving a File Set

To save a file set for use in future multi-file searches, choose **Save this File Set** from the File Sets pop-up menu. The IDE displays the Save File Set dialog box shown in Figure 6.13 on page 189.

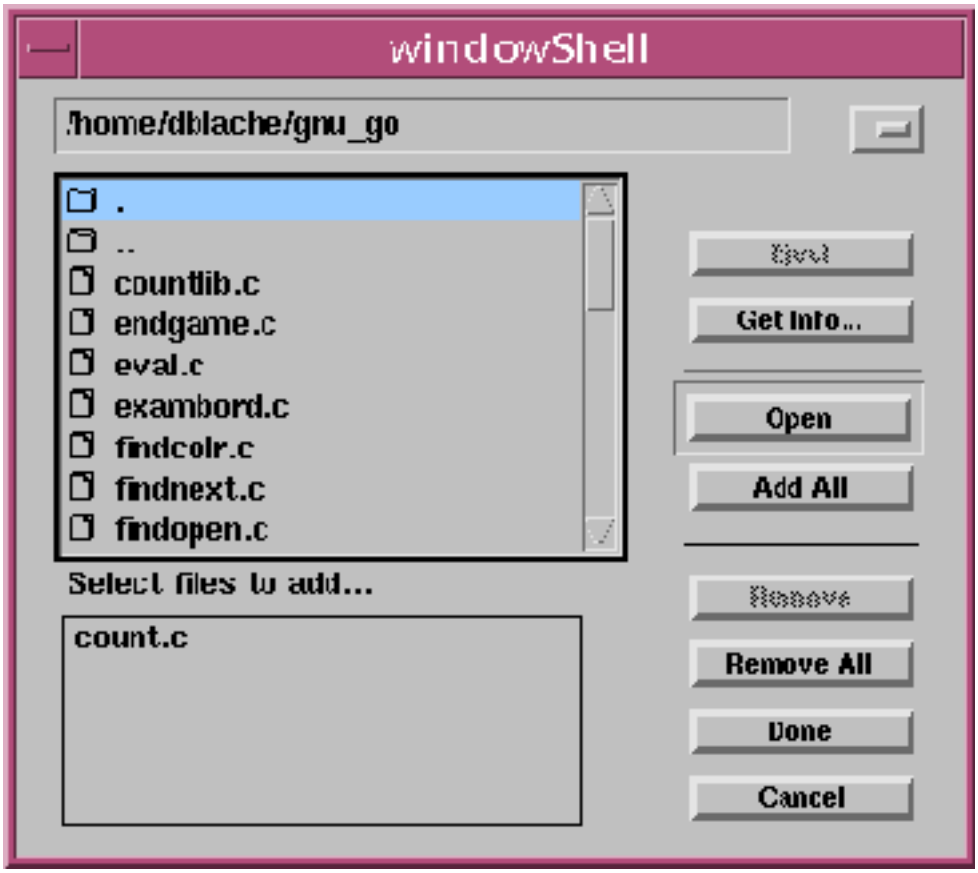
Name the file set by entering a name in the Save File Set As text field.

You can choose the scope of projects which can use this file set. There are two scopes to choose from: **Specific** and **Global**. Click the radio button next to the scope you want to use with this file set.

If you plan to use this file set only with the current project, click **Specific to this project**. The CodeWarrior IDE stores the file set in the project.

If you think you'll use this file set with other projects, click **Global, for all projects**. The CodeWarrior IDE stores the file set in its preferences file, so all projects (even existing projects) can use it.

Figure 6.12 The Select Files To Search dialog box (Solaris)



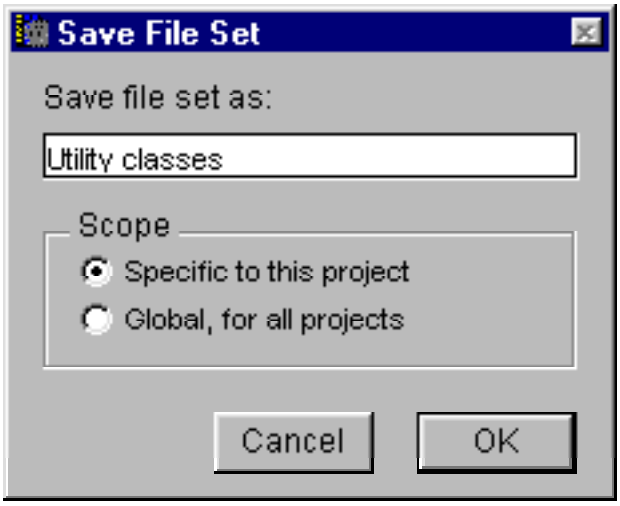
After making your selection and naming the file set, click the **OK** button. If you change your mind and don't want to save the file set, click **Cancel**.

Removing a File Set

To remove a previously-saved file set, choose **Remove a file set** from the File Sets pop-up menu in the Find dialog box. The CodeWarrior IDE displays the dialog box shown in Figure 6.14.

Select the file set you want to remove, then click the **Remove** button. The CodeWarrior IDE removes the file set so that future searches do not use the deleted file set. When you are finished removing file sets, click the **Done** button to return to the Find dialog box. If you change your mind about removing the file set, click **Cancel** instead.

Figure 6.13 The Save File Set dialog box



Controlling Search Range in Multiple Files

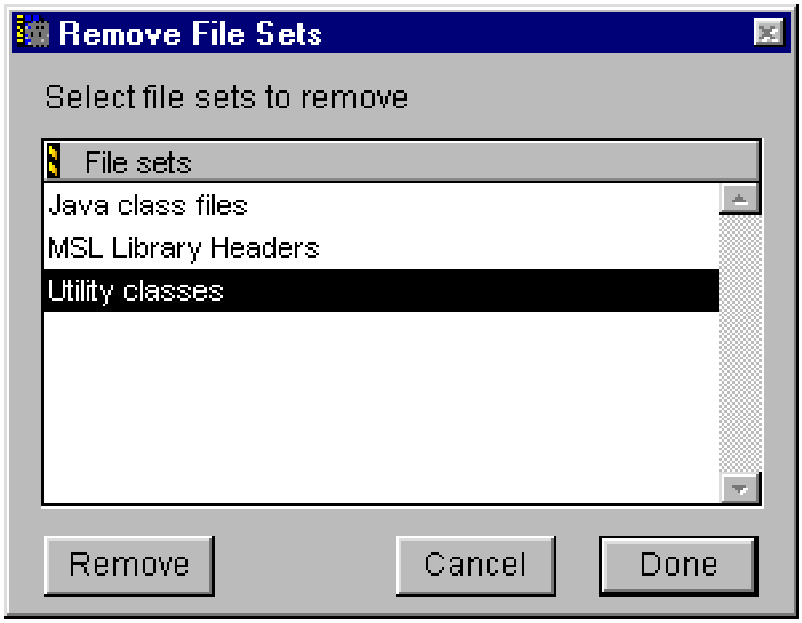
The CodeWarrior editor lets you search any number of files for a string. The files can be in the current project or any text file on disk. If you frequently search a particular set of files, just save that set and restore it later.

You can choose to stop searching at the end of each file or to search all files without stopping.

To treat all the files in the file set as one large file, turn off the Stop at End of File checkbox. When the editor reaches the end of one file, it starts searching the next file until the selected text is found. The editor beeps when it reaches the end of the last file to search. After text is found, you may resume searching for the next occurrence using the **Find**, **Find Next**, or **Find Previous** menu commands.

To search each file individually, enable the Stop at End of File checkbox. The editor beeps when it reaches the end of a file. The arrow to the left of the file set indicates the file the editor is currently searching.

Figure 6.14 Remove File Sets dialog box



You must choose **Find in Next File** from the Search Menu or use the **Find in Previous File** command to continue the search. To start the search from a particular file, just select the file and click in the column to its left.

Windows The Find in Previous File command is available by pressing the key binding Ctrl-Shift-T.

Mac OS The Find in Previous File command is available by pressing the Shift key while using the Search Menu.

After choosing your option, proceed just as you would if you were searching only one file.

TIP: You can search in the previously-searched file in the file list by using the **Find in Previous File** command. This effectively al-

lows you to go backwards in your search into previously-searched files.

To learn more about text searching, see "Searching for Selected Text" on page 172, or "Finding and Replacing Text in Multiple Files" on page 181.

Using Regular Expressions (`grep`)

A regular expression is a text substring that is used as a mask for comparing text in a file. Regular expressions can be formed from single characters or from more complicated strings. When the regular expression is compared with the text in your file, the CodeWarrior IDE analyzes whether the text matches the regular expression you have entered.

This section discusses special operators in regular expressions that the CodeWarrior IDE recognizes and how they can be used to find and replace text. CodeWarrior's regular expressions are similar to UNIX's `grep` commands.

NOTE: Make sure the **Regexp** checkbox is selected in the Find dialog box.

This section consists of the following topics:

- Special Operators
- Using Regular Expressions

Special Operators

The following characters have special meanings based upon their placement in the regular expression. For more information, see "Using Regular Expressions" on page 193.

MetaCharacters	Description
"."	The **match-any-character operator** matches any single printing or non-printing character except `newline` and `null`.
"*"	The **match-zero-or-more** operator repeats the smallest preceding regular expression as many times as necessary (including zero) to match the pattern.
"+"	The **match-one-or-more** operator repeats the preceding regular expression at least once and then as many times as necessary to match the pattern.
"?"	The **match-zero-or-one** operator repeats the preceding regular expression once or not at all.
"\n"	The **back-reference** operator is used in the replace string to refer to a specified group in the find string. Each group must be enclosed within parentheses. The digit *n* must range between 1 and 9. The number identifies a specific group, starting from the left side of the regular expression.
"|"	The **alternation** operator matches one of a choice of regular expressions. If you place the alternation operator between any two regular expressions, the result matches the largest union of strings that it can match.

MetaCharacters	Description
"^"	**The match-beginning-of-line** operator matches the string from the beginning of the string or after a newline character. When it appears within brackets the "^" represents a "not" action.
"$"	**The match-end-of-line** operator matches the string either at the end of the string or before a newline character in the string.
[...]	**List** operators enable you to define a set of items to use as a match. The list items must be enclosed within square brackets. Note that you cannot define an empty list.
(...)	**Group** operators define sub-expressions that can be used elsewhere in the regular expression as a single unit.
"-"	The **range** operator defines the characters that fall between the start and ending characters within the list.

Using Regular Expressions

You can create powerful regular expressions to search for text and perform replace operations on found text. This section will discuss the following topics:

- Matching simple expressions
- Matching any character
- Repeating expressions
- Grouping expressions
- Choosing one character from many
- Matching the beginning or end of a line
- Using the Find string in the Replace string
- Remembering sub-expressions

- References

To give you a better idea of how regular expressions can improve search and replace functions, look over the following example code. Each concept discussed will refer to the example code shown in Listing 6.1.

Listing 6.1 Example Code

```
#include <iostream>

#define  var1  10;
#define  var2  20;

using namespace std;

int result = 0;

int main(void)
{
  cout << “This example provides information about” << endl;
  cout << “the use of regular expressions in the” << endl;
  cout << “CodeWarrior IDE.” << endl << endl;
  cout << “Refer to this code when reading the” << endl;
  cout << “sections in the manual dealing with” << endl;
  cout << “the use of regular expressions.” << endl << endl;
  cout << “The value of var1 is ” << var1;
  cout << “ out of 50, and” << endl;
  cout << “the value of var2 is ” << var2;
  cout << “ out of 50.” << endl;
  cout << “$” << var1;
  cout << “ + $” << var2;
  cout << “ = $” <<;
  result += var1;
  result += var2;
  cout << result;
  cout << endl << endl;
  return 0;
}
```

Matching simple expressions

Most characters match themselves. For example, "`a`" matches all occurrences of the letter "a" in the code. The only exceptions are called special characters: the asterisk (*), plus sign (+), backslash (\), period (.), caret (^), square brackets ([and]), dollar sign ($), and ampersand (&). To match a special character, precede it with a backslash, like this: `\*`

For example, refer to the example code shown in Listing 6.1 on page 194. If you want to find every occurrence of the letter "`m`" in the code, you would type `m` into the Find text box. The CodeWarrior IDE will match this regular expression with the "`m`" in `iostream`. Additional searches will match with the "`m`" in `namespace`, `main(void)`, `example`, `information`, and `manual`.

If you want to find every occurrence of a dollar sign in the code, you would type `\$` in the Find text box. The backslash tells the IDE to interpret the dollar sign as a normal character rather than a special character. The IDE will match the regular expression with the first dollar sign in the following line from Listing 6.1:

```
cout << "$" << var1;
```

Additional searches will match with the second and third dollar signs in the next two lines.

Matching any character

A period (.) matches any character except a newline character. Use the period when you want to be more flexible in your search.

For example, refer to the example code shown in Listing 6.1 on page 194. If you want to find four-character expressions in the code that begin with `var`, type `var.` in the Find text box. The period following the `var` tells the IDE to look through the code for `var` immediately followed by a single character. The IDE will match this regular expression with `var1` and `var2` in the code.

Repeating expressions

The asterisk and plus sign are two special operators that allow you to "repeat" expressions in your search string:

- A regular expression followed by an asterisk (*) matches *zero* or more occurrences of that regular expression. If there is any choice, the editor chooses the longest, left-most matching string in a line.
- A regular expression followed by a plus sign (+) matches *one* or more occurrences of that regular expression. If there is any choice, the editor chooses the longest, left-most matching string in a line.
- A regular expression followed by a question mark (?) matches *zero* or *one* occurrences of that regular expression. If there is any choice, the editor chooses the left-most matching string in a line.

For example, refer to the example code shown in Listing 6.1 on page 194. Suppose you type `s*ion` in the Find text box, The `s*` tells the IDE to match zero or more occurrences of the letter `s` just before the occurrence of `ion`. In Listing 6.1, this regular expression will match with the `ion` in `information` and `sections` from the following lines:

```
cout << "This example provides information about" << endl;
cout << "sections in the manual dealing with" << endl;
```

The same regular expression will also match with the `ssion` in `expressions` from the following lines:

```
cout << "the use of regular expressions in the" << endl;
cout << "the use of regular expressions." << endl << endl;
```

If you want to find matches that have at least one letter `s` preceding `ion`, you would type `s+ion` in the Find text box. The plus sign tells the IDE to match at least one occurrence of the letter `s` just before the occurrence of `ion`. This regular expression will match with the `ssion` in `expressions` from Listing 6.1. Notice that `s+ion` does

not match with the `ion` in `information` or `sections`, since there is not at least one letter `s` preceding `ion` in either case.

If you want to find expressions in the code that contain the number zero, followed by one period or no periods at all, you would type `0\.?` in the Find text box. The backslash tells the IDE to treat the period as a normal character, and the `?` special operator acts on the normal period character. This expression will match with each occurrence of the number zero in Listing 6.1. This regular expression will also match with the `0.` from the following line:

```
cout << " out of 50." << endl;
```

As shown in these examples, the asterisk and plus sign usually refer to a single character. However, it is possible to refer to more than one character at a time by grouping expressions. See the following section, Grouping expressions, for more information.

Grouping expressions

If an expression is enclosed in parentheses (`(` and `)`), the editor treats it as a single unit and applies any asterisk (*) or plus (+) to the whole expression.

For example, refer to the example code shown in Listing 6.1 on page 194. If you want to find expressions in the code that match `is`, you could simply type `is` in the Find text box. However, you could also use `( i)s` as the regular expression. Notice that `( i)s` tells the IDE to look for the letter `s` preceded by both a space and the letter `i`. Whereas `is` matches the `is` within `This`, `this`, and `is`, `( i)s` will match only with `is` in the following two lines from Listing 6.1:

```
cout << "The value of var1 is " << var1;
cout << "The value of var2 is " << var2;
```

Choosing one character from many

A string of characters enclosed in square brackets (`[]`) matches any one character in that string. To match any character that is *not* in the

string enclosed within the square brackets, precede the enclosed expression with a caret (^) like this: `[^abc]`

For example, refer to the example code shown in Listing 6.1 on page 194. If you want to find expressions in the code that contain the letters x, y, or z, you would type `[xyz]` in the Find text box. The IDE will match this regular expression with the letter x in `example` and `expressions`. If instead you want to find expressions in the code that do *not* contain the letters x, y, or z, type `[^xyz]` in the Find text box. The IDE will match this regular expression with every character in the example code except the letter x in `example` and `expressions`.

Placing a minus sign (-) within square brackets indicates a range of consecutive ASCII characters. For example, `[0-9]` is the same as `[0123456789]`. If the minus sign is the first or last character within the enclosed string, that minus sign loses its special meaning and is treated as an ordinary character. For example, `[-bc]` represents the minus sign and the letters b and c.

If a right square bracket is immediately after a left square bracket, it does not terminate the string but is considered to be one of the characters to match. For example, `[]0-9]` tells the IDE to search for the right square bracket as well as any digit in the code. If any special character, such as backslash (\), asterisk (*), or plus sign (+), is immediately after the left square bracket, that special character is treated as an ordinary character. For example, `[.]` tells the IDE to search for periods in the code.

You can use square brackets in a similar manner as parentheses. The IDE will treat the information in the square brackets as a single unit. For example, `[bsl]ag` tells the IDE to search for occurrences of `bag`, `sag`, or `lag` in the code. Typing `[aeiou][0-9]` in the Find text box tells the IDE to search for a vowel followed by a number, such as `a1`.

Matching the beginning or end of a line

You can specify that a regular expression match only the beginning or end of the line.

- If a caret (^) is at the beginning of the entire regular expression, it matches the beginning of a line.
- If a dollar sign ($) is at the end of the entire regular expression, it matches the end of a line.
- If an entire regular expression is enclosed by a caret and dollar sign (`^like this$`), it matches an entire line.

For example, refer to the example code shown in Listing 6.1 on page 194. If you want to find expressions in the code that match `cout` and occur only at the beginning of a line, you would type `^([ \t]*cout)` in the Find text box. The `[ \t]*` in the regular expression allows zero or more spaces and tabs to precede `cout`, as is the case with the code shown in Listing 6.1.

Using the Find string in the Replace string

You can include the contents of the Find string in the Replace string by using an ampersand (`&`) in the Replace string. For example, refer to the example code shown in Listing 6.1 on page 194. Suppose the Find string is `var[0-9]` and the Replace string is `my_&`. The editor will match the Find string with `var1` and `var2` in the code. Clicking the Replace button in the Find dialog box will replace `var1` with `my_var1` or `var2` with `my_var2`.

To use an ampersand in the Replace without any special meaning, use `\&`. An ampersand has no special meaning in the Find string.

Remembering sub-expressions

You can remember sub-expressions of a regular expression in a Find string and recall those sub-expressions in the Replace string. You can create up to nine remembered sub-expressions for each Find string. Each sub-expression must be enclosed within parentheses. To recall these sub-expressions when typing the replace string, use *n*, where *n* is a digit that specifies which sub-expression to recall. Determine *n* by counting sub-expressions from the left side of the Find string.

For example, refer to the example code shown in Listing 6.1 on page 194. If you want to change `#define` declarations into `const` declarations, you could perform a search for `#define` and manually

change the line of code to a `const` declaration. However, you can take advantage of remembered sub-expressions to perform the same process automatically. Begin by typing the following regular expression in the Find text box:

```
\#define[ \t]+(.+)[ \t]+([0-9]+);
```

This regular expression tells the IDE to search for the following string, in the exact order given: `#define`, one or more spaces or tabs, one or more characters, one or more spaces or tabs, one or more digits, and a semicolon. Starting from the left side of the regular expression, the first sub-expression is `(.+)` and the second sub-expression is `([0-9]+)`. These two sub-expressions are recalled in the following Replace string:

```
const int \1 = \2;
```

The `\1` refers to the first sub-expression and the `\2` refers to the second sub-expression from the Find string. These two sub-expressions recall the variable name and its value from the original `#define` declaration. Notice that the replace string changes the `#define` declaration into a `const` declaration by using references to the two sub-expressions. Thus, the editor will find `#define var1 10;` and change it into `const var1 = 10;` in the code shown in Listing 6.1. It will change the next `#define` statement in the same manner.

References

For a comprehensive book on using regular expressions, get *Mastering Regular Expressions*, by Jeffrey E.F. Friedl, published by O'Reilly & Associates, Inc., from your local bookstore.

7

Browsing Source Code

This chapter describes CodeWarrior's class browser, a tool you use to examine your project source code from various perspectives.

Browser Overview

This chapter gives you a full description of the CodeWarrior IDE browser. The browser lets you decide what code is important to look at, and lets you get to that code quickly and easily.

The CodeWarrior IDE browser creates a database of all the symbols in your code, and provides you with a user interface to access the data quickly and easily, regardless of language.

Historically, programmers have used browsers primarily with object-oriented code, but the CodeWarrior IDE browser works with both procedural and object-oriented code. It works with most compilers, including C, C++, Pascal, and Java.

To help you understand the browser, we're going to look at it from three perspectives: high-level architecture, user interface, and functionality. The topics in this chapter include:

- Understanding the Browser Strategy—what the browser is and what it does from a high-level perspective
- Guided Tour of the Browser—what you see when you work with the browser
- Using the Browser—how to use the browser effectively

The rest of this section shows you how to activate the browser so that you can start using it.

Activating the Browser

To learn more about how to activate the browser, refer to:

- "Configuring IDE Overview" on page 235 to learn how to display the dialog box containing the option that activates the browser
- "Choosing Target Settings" on page 293 for an overview of setting target-specific IDE preferences, including browser activation, in the IDE
- "Activate Browser" on page 306 for details about the target-specific preference settings that activate the browser.

When the browser is activated, the compiler generates the browser database information.

For more information on browser settings and options, see "Setting Browser Options" on page 225.

To learn how to use context pop-up menus, see "Context Pop-Up Menu" on page 208.

Understanding the Browser Strategy

When the browser is activated, the CodeWarrior IDE compilers generate a database of information about your code. This database includes data not only about your code, but also about the relationships between various parts of your code, such as inheritance hierarchies.

The browser is a user interface that allows you to sort and sift through this information in ways that suit your needs.

Like any good database access program, the browser does not dictate how you should look at your information. It gives you a variety of tools to suit your working style.

There are three principal ways of looking at the information available to you in the browser:

- Catalog View—a comprehensive view of all data

- Browser View—a class-based view
- Hierarchy View—an inheritance-based view

These sections take a brief look at each option. "Guided Tour of the Browser" on page 207 discusses the user interface in detail.

The browser also implements instant access to information. By right-clicking (Windows) or clicking and holding the mouse (Mac OS and Solaris) on any symbol for which there is information in the database, you get instant access to related source code. "Context Pop-Up Menu" on page 208 discusses this feature.

In addition, the browser gives you one more approach to deciding how you should view data. You decide the scope of the view. You may want to look at data in all your classes, or you may wish to focus on one class.

Within the browser and hierarchy views, you can look at multiple classes or single classes. Table 7.1 summarizes the various major choices you have when using the browser

Table 7.1 Browser viewing options

viewing style	wide focus	narrow focus
comprehensive	catalog	not applicable
class-based	multi-class browser	single-class browser
inheritance-based	multi-class hierarchy	single-class hierarchy

The browser-related menu commands in the Window Menu—**Browser Catalog Window**, **Class Hierarchy Window**, and **New Class Browser**—display wide-focus views. Once you have the wide view, you can focus on a particular class.

No matter what viewing style or focus you happen to be in at any given moment, the browser has simple and intuitive mechanisms for switching to another kind of view.

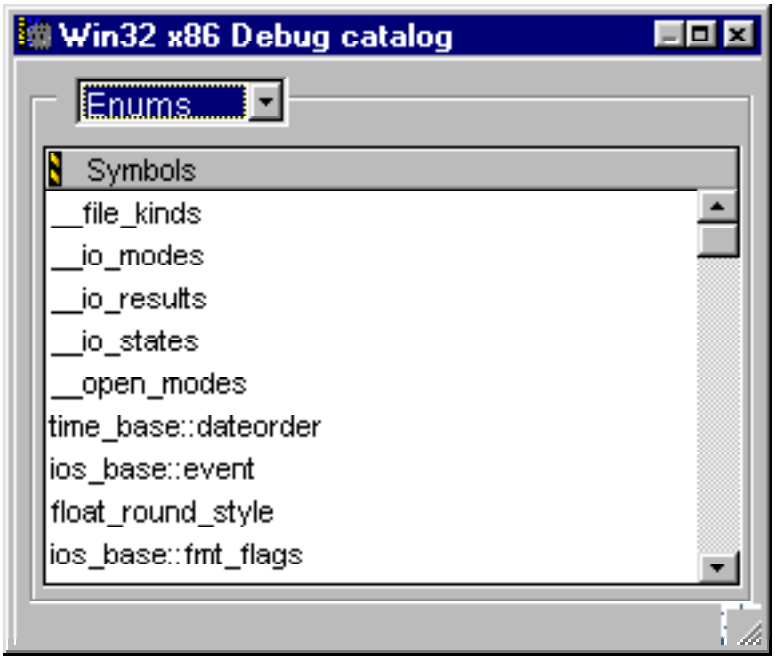
Catalog View

The catalog view lets you see all of your data sorted by category into alphabetical lists. Figure 7.1 shows a catalog view, with the various categories shown in the pop-up menu.

You select the particular category you want to examine. You can focus on classes, constants, enumerations, routines, global variables, macros, routine templates, and type definitions.

See "Catalog Window" on page 209 for details on the catalog window interface.

Figure 7.1 A catalog view



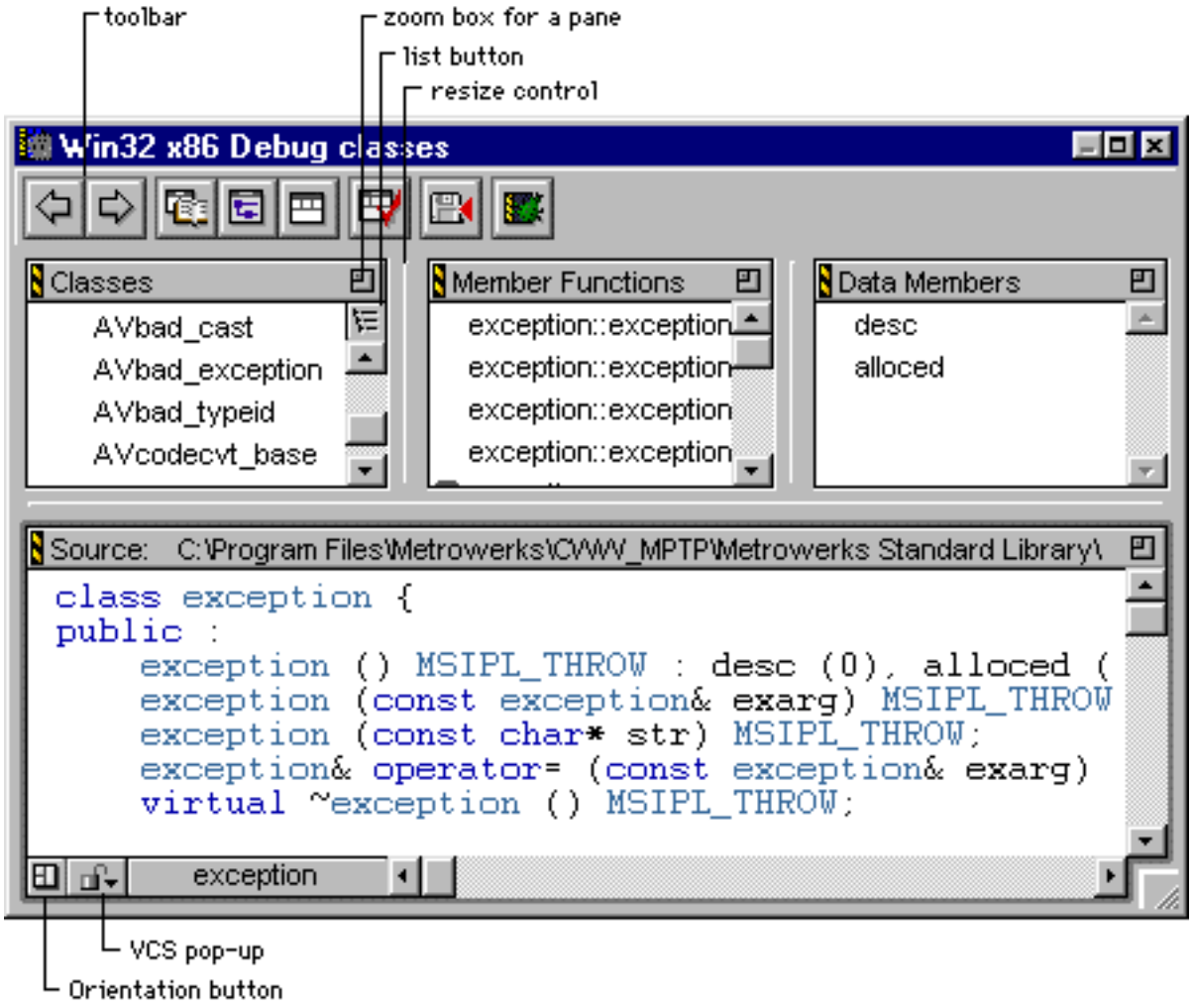
Browser View

The browser view is like a traditional class browser. You use this view to look at your data from a class-oriented perspective. Figure 7.2 shows what a multi-class browser view looks like.

In the browser view, you have a list of classes. For the selected class in that list, you see all of its member routines and its data members. When you select an item, the source code related to that item appears in the Source code pane.

See "Multi-Class Browser Window" on page 211 and "Single-Class Browser Window" on page 217 for details on the browser view interface.

Figure 7.2 A browser view (multi-class)



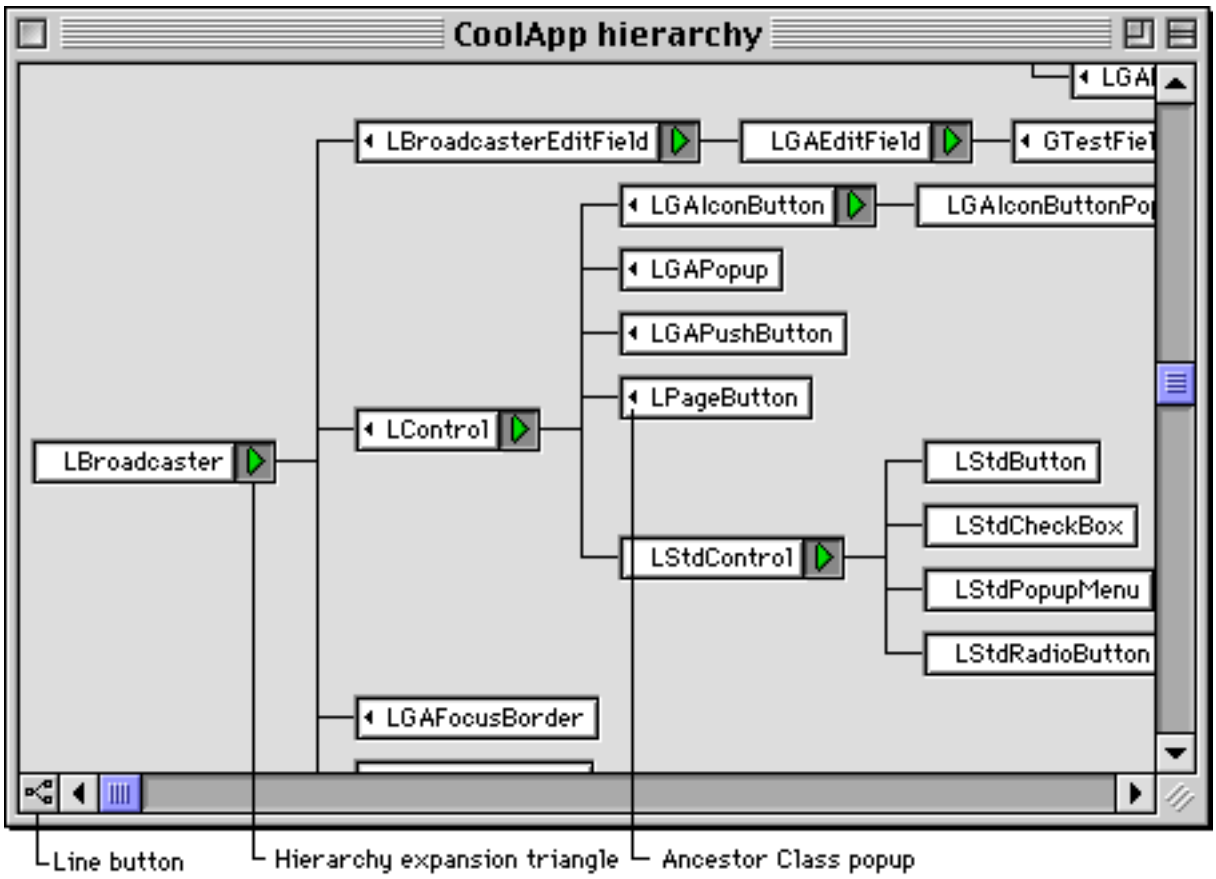
NOTE: The class browser view has a toolbar. To learn how to use and customize the toolbar, see "Customizing Toolbars" on page 280.

Hierarchy View

The hierarchy view is a graphical view of your class hierarchy. You use this view to understand or follow class relationships. Figure 7.3 illustrates a multi-class hierarchy view for some classes.

The hierarchy view gives you a real feel for the way your classes are connected with each other. You can expand and collapse a hierarchy at will.

Figure 7.3 A hierarchy view



See "Multi-Class Hierarchy Window" on page 219 and "Single-Class Hierarchy Window" on page 221 for details on the hierarchy view interface.

Guided Tour of the Browser

At first glance, the browser interface is quite complicated. There are multiple windows filled with controls and information. However, there are really only three kinds of views: catalog, browser, and hierarchy.

In addition, context pop-up menus are a fundamental and vital feature of the browser interface. Use these pop-up menus to make navigating code simple. Pop-up menus are available for any symbol for which there is data in the browser database. They describe various actions that you can perform on the selected item.

For object-oriented code (member functions only), you have the opportunity to see the declaration or definition of any routine with that name. You can also open up a symbol browser that lists every implementation of the routine. Depending upon the nature of the symbol (class name, routine name, enumeration, and so forth), the pop-up menu lists different destinations appropriate for the item. This gives you instant access to the source code related to any symbol.

This section examines each window used by the browser, its controls, and the Context Pop-Up Menu. The sections are:

- Context Pop-Up Menu
- Catalog Window
- Multi-Class Browser Window
- Single-Class Browser Window
- Multi-Class Hierarchy Window
- Single-Class Hierarchy Window
- Symbol Window

Context Pop-Up Menu

When the browser is active, right-click (Windows) or click and hold (Mac OS and Solaris) on any symbol for which there is data in the browser database. When you do, a pop-up menu appears with a variety of items.

The nature of the items in the pop-up menu depends upon the nature of the symbol you are investigating. Some items may allow you to open browser windows. In every case, one or more items in the menu direct you to a location in code

You can also use the context pop-up menu to perform common debugging tasks. For example, you can use the pop-up menu to set and clear breakpoints in the active editor window.

In effect, every symbol in your code—routine name, class name, data member name, constant, enumeration, template, macro, type definition, and so forth—becomes a hypertext link to a location or multiple locations in your source code.

For example, if you right-click (Windows) or click and hold (Mac OS and Solaris) on a class name, you can open the class declaration, a Single-Class Browser Window, or a Single-Class Hierarchy Window. If you right-click (Windows) or click and hold (Mac OS and Solaris) on a routine name, you get different choices, as shown in Figure 7.4.

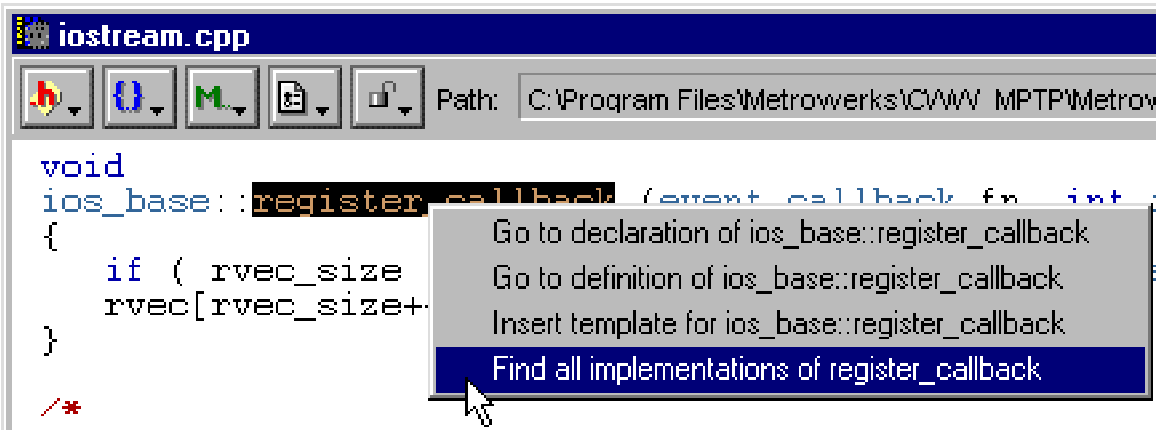
You can even insert a routine template at the location if you wish. See "Browser Display" on page 260 for more information.

Other menus for other kinds of symbols have items of a similar nature.

TIP: To find and enter a browser item for a piece of text you've selected or just entered, right-click (Windows) or click and hold (Mac OS and Solaris) on the text and wait for the Context pop-up menu to appear. The menu will offer a list of matching items. Choosing one of these items will enter that item for you. See

"Completing Symbols" on page 229 for other ways to type browser items.

Figure 7.4 The context pop-up menu for a routine



TIP: The context pop-up feature works not only in browser windows, but also in the CodeWarrior source code Editor! This is a great reason for always having the browser enabled, even if you don't use the browser windows.

Of particular note in the Context Pop-Up Menu for routine names is the *"Find all implementations of"* item. When you choose this item, a Symbol Window opens in the browser.

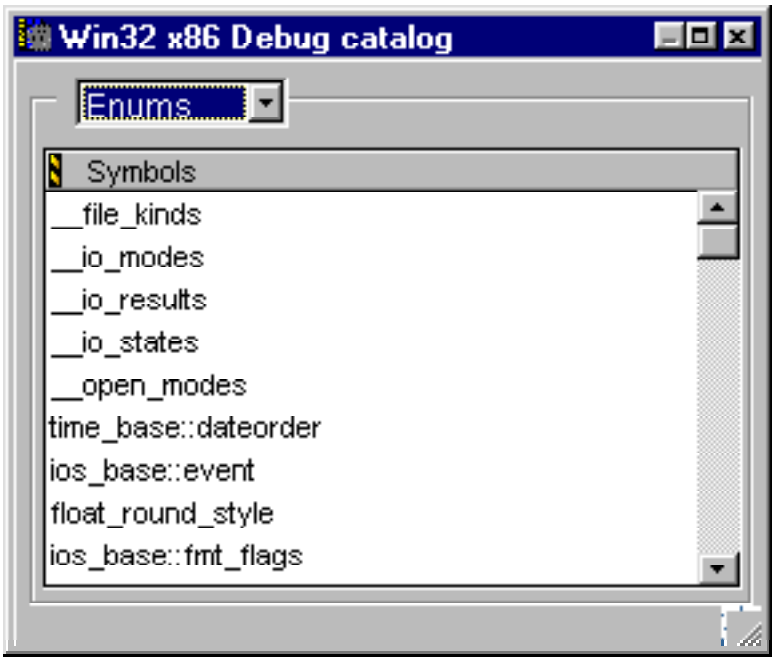
Catalog Window

The Catalog window displays browser data sorted by category into alphabetical lists. Choose **Browser Catalog Window** from the Window Menu to display the Catalog window. Figure 7.5 shows the catalog window.

The items in this window are:

- Category pop-up menu
- Symbols pane

Figure 7.5 A catalog window



Category pop-up menu

The Category pop-up menu in the Catalog window controls the current type of information on display in the Symbols pane. The currently selected item is highlighted (Windows) or checked (Mac OS and Solaris).

Symbols pane

The Symbols pane displays every item in the browser database that is a member of the currently selected category. The items are listed alphabetically.

NOTE: Routines are listed alphabetically by *routine* name, but the class name appears first. As a result, it may appear that the routines are not listed alphabetically.

Multi-Class Browser Window

The Multi-Class Browser window (Figure 7.2 on page 205) gives you a class-based view of every class in the browser database. The window has several panes displaying lists of information. Choose **New Class Browser** from the Window Menu to display the Multi-Class Browser window.

The Multi-Class Browser window also has a zoom feature for each pane within the window. Simply click the zoom box in any pane to resize it to fill the entire window, hiding the other panes. Clicking the same zoom box again restores the pane to its previous state.

TIP: The class browser view has a toolbar. To learn how to use and customize the toolbar, see "Customizing Toolbars" on page 280.

The Classes pane, Member Functions pane, and Data Members pane are lists of their respective data. Clicking within a particular pane makes it the active pane. You can also use the Tab key to rotate through all the panes (except for the Source pane).

TIP: Using the Tab key can be mildly hazardous to your source code. If the Source pane is active and you press the Tab key, you enter a tab into your source code. Once you are in the Source pane, you can't press the Tab key to get out of it.

Windows Use the mouse to select a different pane.

Mac OS and Solaris Use the mouse or press Option-Tab to move to a different pane. A grey focus box outlines the currently active pane.

You can click an item in any list to select it, or navigate through the items in the active list by typing or using the arrow keys. You can type in a name, and as you type the selection changes.

Mac OS and Solaris Use the Control-Tab key combination to cycle through list items alphabetically.

The items in this window include:

- Browser Toolbar
- Pane Zoom Box
- Resize bar
- Classes pane
- List button
- Member Functions pane
- Data Members pane
- Source pane
- Orientation button
- VCS Pop-up button
- Open File button
- Identifier icon

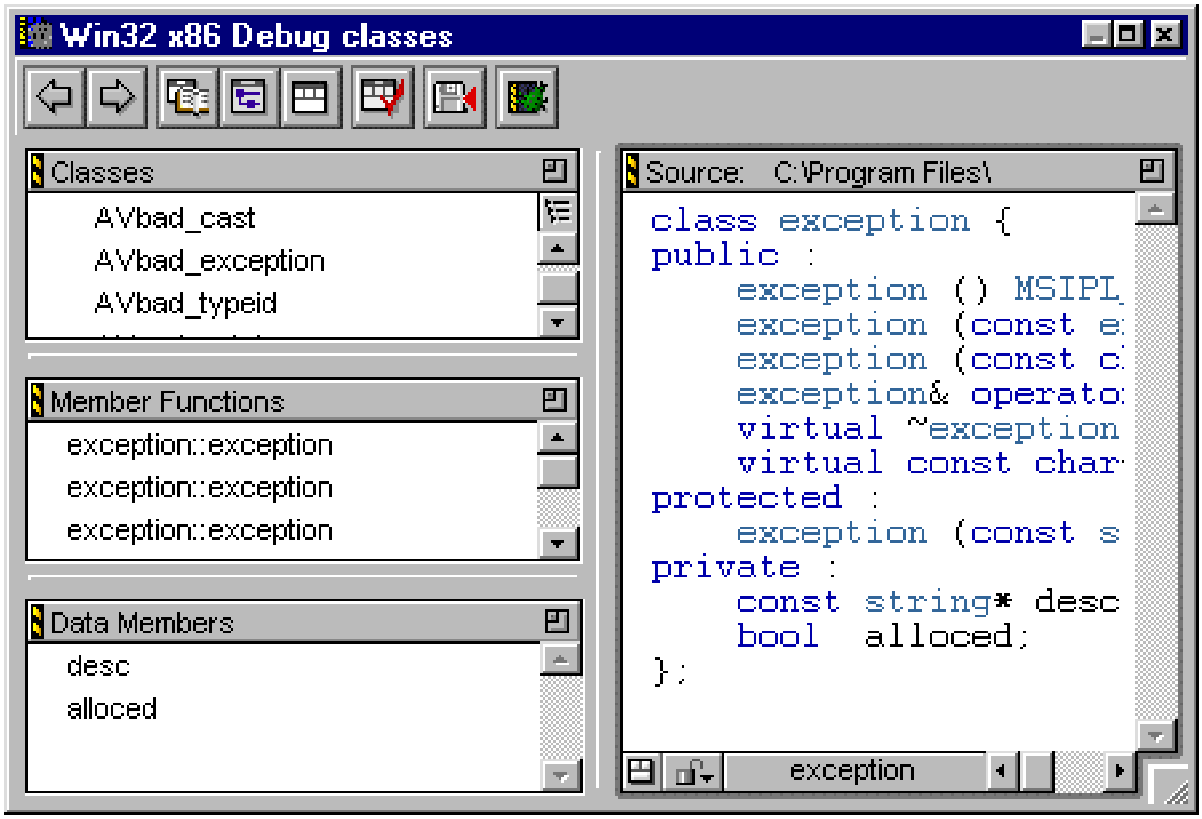
Browser Toolbar

The toolbar provides easy single-click access to many CodeWarrior commands. The buttons that appear in the Browser window's toolbar include:

Button	Description
	Go Back
	Go Forward
	Show Catalog window
	Show Hierarchy window
	New Class Browser
	Save Default window

Button	Description
	Save file
	Switch to MW Debugger

Figure 7.6 The multi-class browser in vertical orientation



Pane Zoom Box

Use the Pane Zoom box to enlarge a pane to fill the Browser window, or shrink it to its original size.

Resize bar

A Resize bar appears between each pair of panes. To resize a pane, click and drag the resize bar next to that pane.

Classes pane

The Classes pane lists all the classes in the browser database.

You can view the list alphabetically or by class hierarchy. You toggle the view by clicking the List button at the top right of the pane, as shown in Figure 7.2 on page 205.

In hierarchy view, tree controls (Windows) or disclosure triangles (Mac OS and Solaris) appear next to class names that have subclasses, as shown in Figure 7.3 on page 206. Click the tree control or disclosure triangle to show or hide subclasses.

TIP: Alt/Option click a tree control or disclosure triangle to open all subclasses at all levels. This is a "deep" disclosure. Ctrl/Command click to open a single level of subclass in a class and all of its siblings at the same level. This is a "wide" disclosure.

When you select a class in the Classes pane, the Multi-Class Hierarchy Window selection changes too. It will scroll to the newly selected class if necessary.

List button

The List button at the top right of the Classes pane controls the display of classes. You can toggle between an alphabetical list or a hierarchical list.

Click this button to switch to an alphabetical list.

Click this button to switch to a hierarchical list.

Member Functions pane

The Member Functions pane lists all member functions defined in the currently selected class. This list does not include inherited functions. In a Single-Class Browser Window, you may display inherited member functions.

Constructors and destructors are at the top of the list. After that, all other entries are alphabetical.

Data Members pane

The Data Members pane lists all data members defined in the currently selected class. This list does not include inherited data members. In a Single-Class Browser Window, you may display inherited data members.

The entries in the Data Members pane are alphabetical. If inherited members are displayed, data members are listed by superclass, but alphabetically within each class.

Source pane

The Source pane displays the source code for the currently selected item. If the item is a class, this pane shows the class declaration. If the item is a routine, this pane shows the routine definition. If the selected item is a data member, it shows the data member declaration from the interface file.

The text in the source pane is fully editable.

TIP: If you Alt/Option click an item in the Member Functions pane or the Data Members pane, that item is entered into the Source pane text at the current insertion point. This is a neat way to enter routine calls or variable names into the Source pane.

The path to the file that contains the code on display is shown at the top of the pane.

There are three buttons at the bottom of the pane. The Orientation button modifies the arrangement of panes in the window. the VCS Pop-up button enables revision control in the source file. The Open File button opens the source file containing the code on display.

Orientation button

The Orientation button at the bottom left of the Source pane controls the distribution of panes in the window. The window may be oriented horizontally, as in Figure 7.2 on page 205, or vertically as in Figure 7.6 on page 213.

Click this button to switch to a horizontal orientation.

Click this button to switch to a vertical orientation.

VCS Pop-up button

This pop-up allows you to use revision control with the source file you are viewing. To learn more about how to use this feature, refer to the discussion in "Using Source Code Control with Files" on page 359.

Open File button

The File button displays the name of the file that contains the code on view in the Source pane. Click this button to open the source file. The Open File button uses the CodeWarrior Editor to open the file.

Identifier icon

A routine or data member may have an identifier icon beside its name. The following table describes the icons.

Table 7.2 Browser identifier icons

Icon	Meaning	The member is…
	static	a static member
	virtual	a virtual function that you can override, or an override of an inherited function
	pure virtual	a member function that you must override in a subclass if you want to create instances of that subclass

Single-Class Browser Window

The Single-Class Browser Window gives you a detailed view of one class in the browser database. The window is similar to the Multi-Class Browser Window. Figure 7.7 shows the window.

You can open a single-class browser using several techniques, including:

- Use the Context Pop-Up Menu in the Catalog View when classes or functions are displayed
- Double-click a class name in a Multi-Class Hierarchy Window
- Double-click a class name in a Single-Class Hierarchy Window
- Use the Context Pop-Up Menu in the Multi-Class Browser Window

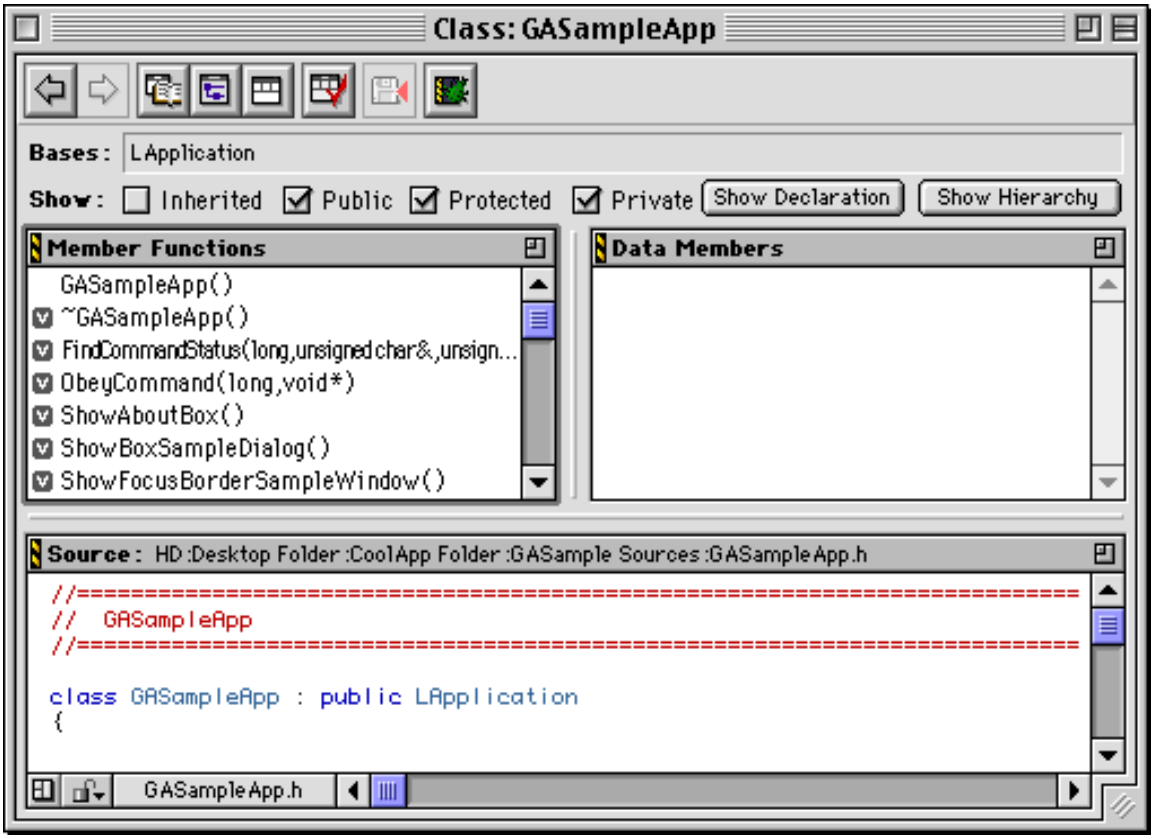
---

TIP: If you select a class in the Classes pane in the Multi-Class Browser Window, you can use the Enter/Return key to open a browser window for the selected class.

The appearance and behavior of this window is similar to the Multi-Class Browser Window. For a discussion of the window in general, see "Multi-Class Browser Window" on page 211. That topic discusses common items in both the Multi-Class Browser window and the Single-Class Browser Window:

- Resize bar
- Member Functions pane
- Data Members pane
- Source pane
- Orientation button
- Open File button
- Identifier icon

Figure 7.7 The single-class browser window



The Single-Class Browser Window also has some unique items. They are:

- Bases text field
- Show checkboxes
- Show Declaration button
- Show Hierarchy button

Bases text field

The Bases text field lists all the immediate base classes for the class on display in the browser window. This list does not include more distant ancestors of the class, only those from which it is a direct and immediate descendant.

This field is informational only and cannot be edited.

Show checkboxes

The checkboxes in the Show section of the Single-Class Browser Window control what kinds of data are displayed in the window. You may turn any individual item on or off. The possibilities are: **Inherited**, **Public**, **Protected**, and **Private**.

The choices you make apply to both member functions and data members.

Show Declaration button

When you click the **Show Declaration** button, the class declaration appears in the Source pane.

Show Hierarchy button

If you click the **Show Hierarchy** button, you open a Single-Class Hierarchy Window for the class on display in the Single-Class Browser Window.

Multi-Class Hierarchy Window

The Multi-Class Hierarchy window displays a complete graphical map of the classes in the browser database. Each class name appears in a box, and related classes are connected to each other by lines. Choose **Class Hierarchy Window** from the Window Menu to display the Multi-Class Hierarchy window. Figure 7.8 shows the window.

Use the arrow keys to change the selected class "geographically." The up and down keys work on siblings. The left and right keys work on ancestors and descendants.

You can type ahead to change the selection. Use the Tab key to change the selected class alphabetically.

TIP: If you select a class in the Classes pane in the Multi-Class Browser Window, the selection in the Multi-Class Hierarchy Window changes too.

If you double-click a class entry, or select the entry and press the Enter/Return key, you open a Single-Class Browser Window for that class.

In addition to the entry for each class, this window has three items:

- Line button
- Hierarchy expansion triangle
- Ancestor Class pop-up menu

Line button

The Line button controls the appearance of the lines that connect related classes. You can toggle between diagonal lines and straight lines. The choice is entirely aesthetic.

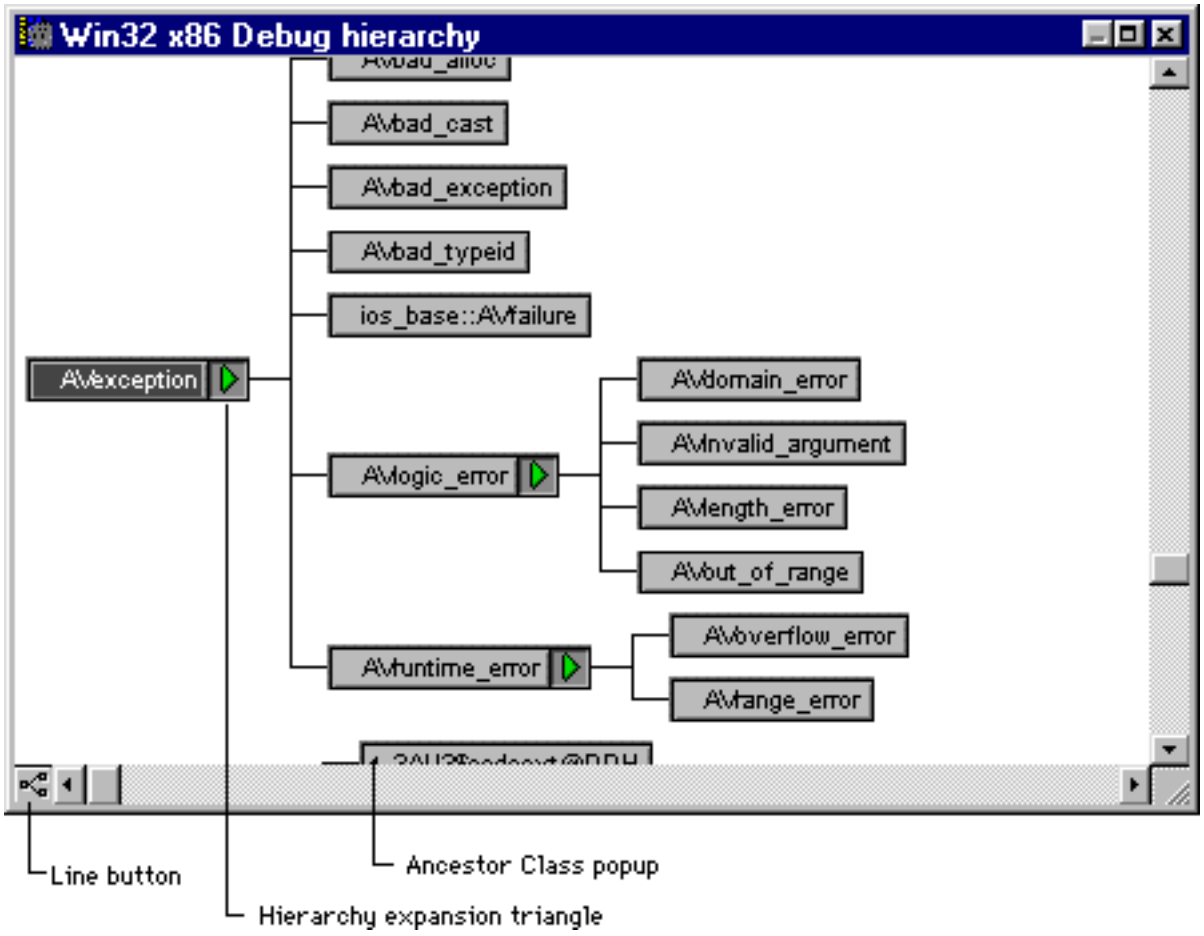
Hierarchy expansion triangle

The Hierarchy expansion triangle controls the display of subclasses.

If you click this triangle, the next level of subclasses appears or disappears. To be more precise, the expanded state restores to what it was the last time this class was open.

NOTE: Alt/Option click a disclosure triangle to open all subclasses at all levels. This is a “deep” disclosure. Ctrl/Command click to display subclasses for a class and all of its siblings. This is a “wide” disclosure. Ctrl-Alt-click (Windows) or Command-Option-click (Mac OS and Solaris) to open both wide and deep. You can use Ctrl/Command-Alt/Option-click to expand or collapse an entire map if you click the expansion triangle for a base class that has no ancestors.

Figure 7.8 The Multi-Class Hierarchy Window



Ancestor Class pop-up menu

Click on the Ancestor Class triangle to display the pop-up menu. The menu lists immediate ancestors. When you choose an item in the pop-up menu, you jump to that class in the map. If the item is not currently visible, the computer beeps.

This control appears only for classes that have multiple base classes.

Single-Class Hierarchy Window

The Single-Class Hierarchy Window displays a complete graphical map for a single class in the browser database. The map displays *all*

immediate ancestors of the class, and all of its descendants. (The Multi-Class Hierarchy Window only shows one base class.)

Figure 7.9 shows the window, displaying multiple base classes and subclasses. The underlined class name is the focus of the window.

You can open a single class hierarchy view using several techniques:

- Use the Context Pop-Up Menu in the Catalog Window
- Use the Context Pop-Up Menu in the Multi-Class Hierarchy Window
- Use the Context Pop-Up Menu in the Multi-Class Browser Window
- Use the **Show Hierarchy** button in a Single-Class Browser Window

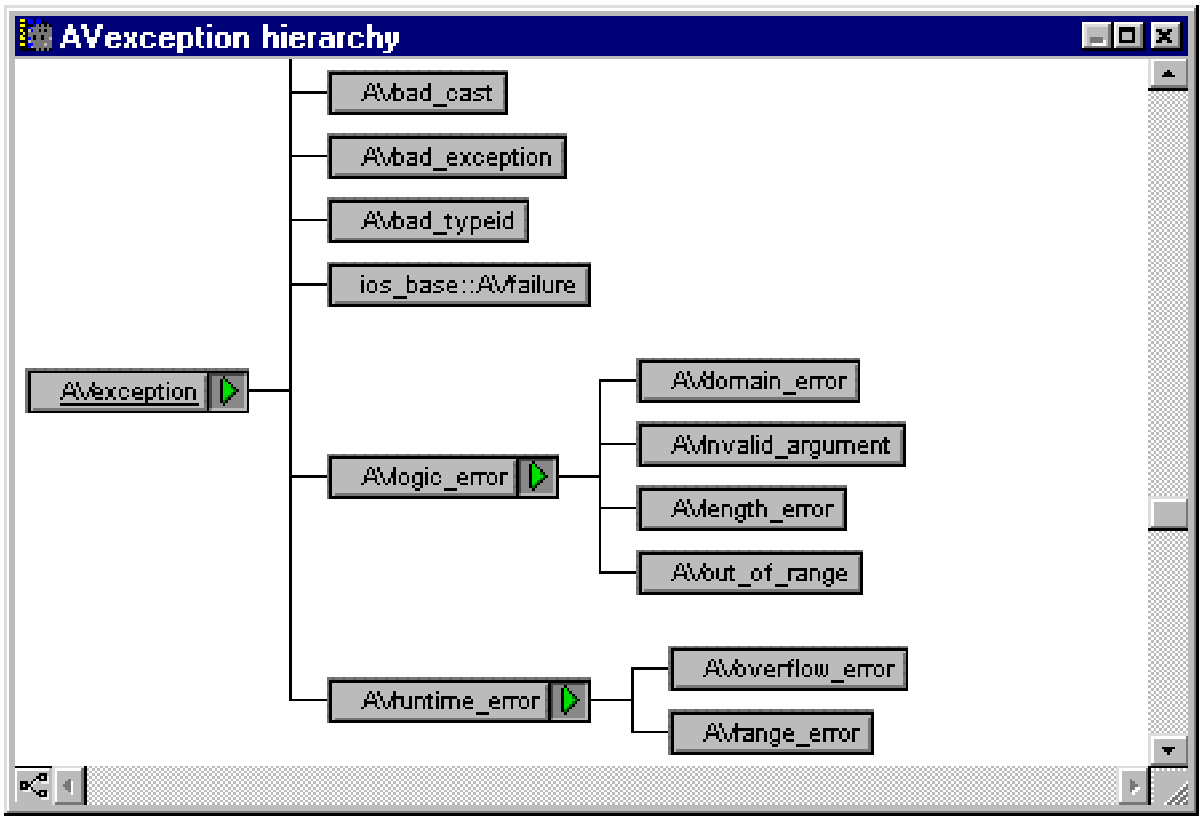
The Single-Class Hierarchy Window is identical to the Multi-Class Hierarchy Window, except that it displays a limited map. For information about how this window behaves, see "Multi-Class Hierarchy Window" on page 219.

Symbol Window

The Symbol window lists all implementations of any symbol that has multiple definitions. Most commonly, these are multiple versions of overridden functions in object-oriented code. However, the Symbol window works for *any* symbol that is multiply defined in the database.

By selecting an implementation in this list, you see its definition in the Source pane. Figure 7.10 on page 224 shows the Symbol window.

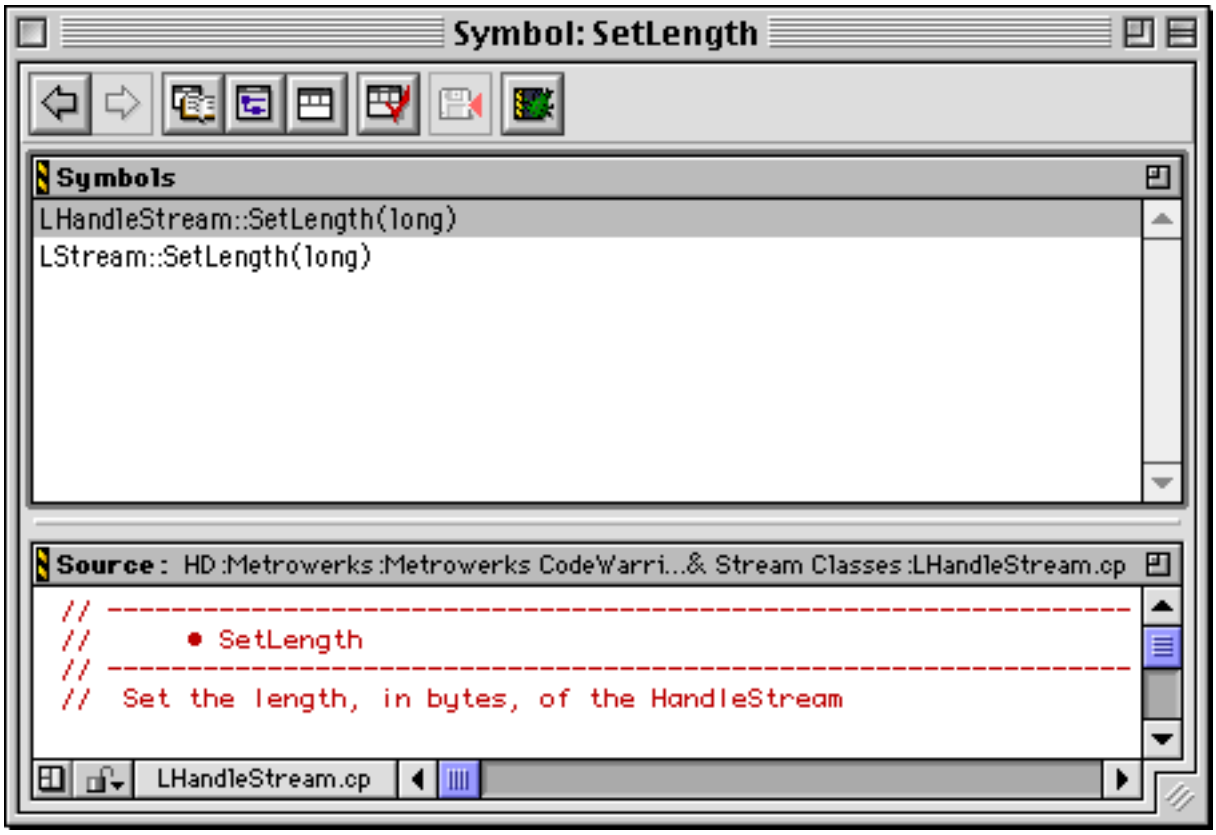
Figure 7.9 The Single-Class Hierarchy window



You open a Symbol window by right-clicking (Windows) or clicking and holding (Mac OS and Solaris) on a symbol name in any browser or editor window for which there is information in the browser database. When you do, a Context Pop-Up Menu appears. If the item has multiple implementations, one item in the pop-up menu will be "*Find all implementations of*." When you choose that item, the Symbol window appears.

TIP: In a Source pane or editor window, Alt/Option-double-click or Ctrl/Command-double-click a function or other symbol name to find all implementations and open the Symbol window without using the pop-up menu.

Figure 7.10 The browser Symbol window



Most of the items in this window are identical to the Multi-Class Browser Window. For a discussion of the window in general, see "Multi-Class Browser Window" on page 211. That topic includes discussion of these items, also found in the Symbol window:

- Resize bar
- Source pane
- Orientation button
- Open File button

This window also has one unique item, the Symbols pane. The Symbols pane lists all versions of a symbol in the database.

When you select an item in the list, that item's definition appears in the Source pane.

Using the Browser

The browser provides multiple paths through the data related to your code. There is no way that this manual can define all possible browser paths through arbitrary source code. What we can do is give you a feel for how to work with the browser, and outline some techniques you use to accomplish common tasks.

Topics in this section include:

- Setting Browser Options
- Identifying Symbols in the Browser Database
- Navigating Code in the Browser
- Browsing Across Subprojects
- Completing Symbols
- Opening a Source File
- Seeing a Declaration
- Seeing a Routine Definition
- Editing Code in the Browser
- Analyzing Inheritance
- Finding Functions That Are Overrides
- MFC Class Viewing (Windows)
- PowerPlant Class Viewing (Mac OS)
- Saving a Default Browser

Setting Browser Options

Browser-related menu items and browser-specific options become available when you activate the browser. See "Activating the Browser" on page 202 for information on how to turn the browser on.

When the browser is on, browser-related menu items are enabled. These are the **Browser Catalog Window**, **Class Hierarchy Window**, and **New Class Browser** items in the Window Menu.

TIP: A quick way to tell whether the browser is enabled is to look in the Window Menu at the browser-related menu items. If they are enabled, the browser is activated.

In addition, there are global IDE options that relate to the browser. You control how various items are colored in browser windows, and the time delay before the Context Pop-Up Menu appears (Mac OS and Solaris).

To tell the IDE to include items from a project's subprojects in its browser windows, see "Cache Subprojects" on page 306.

To learn how to modify these settings, see "Browser Display" on page 260 or "Context Popup Delay" on page 265.

Identifying Symbols in the Browser Database

There is an easy way to know whether or not a symbol is in the browser database without checking to see if a Context Pop-Up Menu appears: use browser coloring. If the browser is activated, symbols that are in the browser database will be displayed in editor and browser windows according to the colors you select. See "Browser Display" on page 260 for more information.

TIP: The factory default color setting is identical for all of the eight types of browser-database symbols. You can choose a different color for each symbol type if you like. However, if you have also enabled syntax coloring for your code, you may find it easier to identify symbols in the browser database using only one or two colors.

Navigating Code in the Browser

There are many ways to move around in code with the browser.

Using the Context Pop-Up Menu

Perhaps most powerful and flexible way to navigate code is to use the Context Pop-Up Menu. You see this menu when you right-click (Windows) or click and hold (Mac OS and Solaris) on any symbol for which there is data in the browser database. This includes class names, routine names, global variables, class data members, and much more. To learn more, refer to "Context Pop-Up Menu" on page 208.

In the Multi-Class Browser Window and Single-Class Browser Window, simply selecting a class, routine, or data member displays the associated code in the window's Source pane.

Go Back and Go Forward

The browser fully supports the **Go Back** and **Go Forward** items in the Search Menu. No matter what views, windows, or code you have looked at, you can always go back to what you had been viewing earlier.

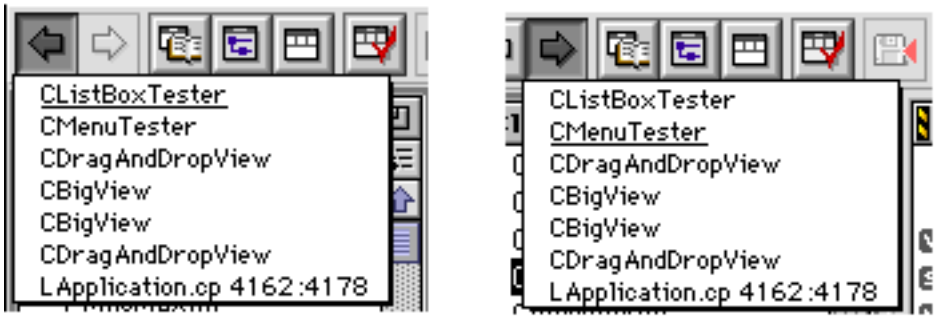
TIP: If you have the **Go Back** and **Go Forward** tools in the toolbar, click and hold on the tool icon to see a pop-up menu of all the locations in the go-back queue. You can jump directly to any view in the queue.

These commands allow you to go backward or forward in a series of changes you made. For example, say you use the Browser to look at your project and make changes to a file. Then, you switch files and make more changes. You may do this many times. You use the Go Back command to go back one or more actions you have performed. Even if you didn't make any changes to the file, but looked at it (or a specific class or method), you can go back to that action. Similarly, once you've gone back, you can use the Go Forward command to return you to where you started.

When you add the Go Back and Go Forward commands to a CodeWarrior toolbar, you can use their associated pop-up menus,

as shown in Figure 7.11. Click and hold on the Go Back or Go Forward toolbar icons to display the pop-up menu.

Figure 7.11 Go Back and Go Forward toolbar buttons



Windows Choose any item from the menu to go to that action. If you choose an action out of sequence, CodeWarrior will go to that action *without* going through any previous action.

Mac OS The underlined item is your current position in the queue. Choose any item from the menu to go to that action. If you choose an action out of sequence, CodeWarrior will go to that action *without* going through any previous action. The CodeWarrior IDE can track up to 100 actions.

NOTE: **Go Back** and **Go Forward** do *not* undo any actions you performed. They allow for a more flexible method of moving around to specific places you have been in the Browser window.

Browsing Across Subprojects

The IDE normally only displays browser items for the current target. To include browser information from the current target's subprojects in the browser's views, turn on subproject caching. For more information, see "Cache Subprojects" on page 306.

Completing Symbols

The IDE has commands to complete your typing for you when you enter the name of an item that the class browser is aware of. Use the **Find symbols with prefix**, **Find symbols with substring**, **Get next symbol**, and **Get previous symbol** keyboard commands to find and choose browser items that match the text you've selected or just entered in a source code file. These commands are only available from the keyboard. They aren't available in the IDE's menus. See "Editor Commands" on page 428 for a description of these keyboard commands.

To enter the name of a browser item that has the same characters as the text you've selected or just typed, use the **Find symbols with substring** keyboard command. To enter the name of a browser item that only has the same first characters, use the **Find symbols with prefix** keyboard command.

After using the **Find symbols with substring** and **Find symbols with prefix** commands, use the **Get next symbol** and **Get previous symbol** commands to search among the browser symbols that match the text you've selected or just entered.

After you've found the browser item you want to enter, press the right arrow key to place the insertion point next to the item and continue typing.

TIP: Another way to find and enter a browser item is to right-click (Windows) or click and hold down the mouse button (Mac OS and Solaris) on the first few characters of the text and wait for the Context pop-up menu to appear. The menu will offer a list of matching items. Choosing one of these items will enter it for you. See "Context Pop-Up Menu" on page 208 for more information.

Opening a Source File

There are some quick methods you can use to open a source file you want to see.

In the Multi-Class Browser Window or Single-Class Browser Window, click the Open File button when portions of the file you wish to see are displayed in the window's Source pane. See "Open File button" on page 216 for more information.

Click on a symbol used in the file to select it. Then use the Context Pop-Up Menu to open the desired file or see a particular routine.

Seeing a Declaration

There are several methods used to see a declaration. The methods vary depending upon the kind of symbol you are investigating.

If you select a class name or data member name in a Multi-Class Browser Window or Single-Class Browser Window, and the declaration appears in the Source pane, double-click the name to open the file that contains the declaration. (If you select or double-click a routine name, you see the definition).

If you click the Show Declaration button in the Single-Class Browser Window, you see a class declaration.

When you right-click (Windows) or click and hold (Mac OS and Solaris) on a name in any window, use the Context Pop-Up Menu to open the declaration. This technique lets you see a routine declaration.

Seeing a Routine Definition

There are several methods you can use to see a routine definition:

- In the Multi-Class Browser Window or Single-Class Browser Window, select the routine in the Member Functions pane. The definition appears in the Source pane. To open the file that contains the definition, double-click the routine.
- Right-click (Windows) or click and hold (Mac OS and Solaris) on the routine name in any editor or browser window. Then use the Context Pop-Up Menu and jump to the particular routine definition.
- Choose the **Go Back** item from the Search Menu.

- Press Alt/Option or Ctrl/Option, then double-click a function name in any source view to open the Symbol Window with all implementations of the function.
- In the Multi-Class Hierarchy Window or the Single-Class Hierarchy Window, right-click (Windows) or click and hold (Mac OS and Solaris) on a class name. Then use the Context Pop-Up Menu to jump to the particular routine definition.

Editing Code in the Browser

Any code visible in a Source pane is fully editable. Locate the definition you wish to work with. When the code appears in the Source pane, use the same techniques you would in any CodeWarrior Editor window.

For more information about the CodeWarrior Editor, see "Source Code Editor Overview" on page 129.

Analyzing Inheritance

Use the Multi-Class Hierarchy Window or the Single-Class Hierarchy Window to analyze inheritance in your source code. Look for the small disclosure triangle on the left of a class name that indicates the class has multiple ancestors. Use the associated Ancestor Class pop-up menu to jump to any ancestor class to study its descendants (or ancestors). Use the Hierarchy expansion triangle to expose or conceal subclasses.

For more information, see "Multi-Class Hierarchy Window" on page 219.

Finding Functions That Are Overrides

Use the Single-Class Browser Window to find functions that are overrides. Turn off the display of inherited functions. This disables the display of unchanged inherited functions. However, it does not turn off functions you have overridden.

Next, look for functions that are marked as virtual with an Identifier icon. Most of these functions are likely to be overrides of inherited

functions, although some could be functions you declared in this class that were not inherited from an ancestor.

To open a Symbols pane window for any routine, right-click (Windows) or click and hold (Mac OS and Solaris) on a routine name, and then choose the "*Find all implementations of*" item. Combined with a hierarchy view, you see precisely who overrides the routine, and where the routines are found in the class hierarchy.

MFC Class Viewing (Windows)

If you want the browser to display MFC classes, you have to include MFC headers in such a way that the compiler sees them.

You should use precompiled headers to speed compilation time. If you simply use the precompiled headers directly, however, the compiler does not see MFC classes and does not generate information for the symbol database.

The way around this problem is to include a renamed copy of the source file used to create the precompiled headers in your project. These source files have a name like `myFile.pch`.

This creates a project-specific precompiled header. By including the source file, CodeWarrior builds the precompiled header from within your project, exposing MFC symbols to the compiler, which generates information for the browser database.

The reason you want to include a renamed copy is to avoid problems if you have multiple projects that use the same precompiled headers. When one project updates the headers, it will be marked as changed for all other projects, which then rebuild the precompiled headers. This defeats the purpose of the precompiled header.

PowerPlant Class Viewing (Mac OS)

If you want the browser to display PowerPlant classes, you have to include PowerPlant headers in such a way that the compiler sees them.

You should use precompiled headers to speed compilation time. If you simply use the precompiled headers directly, however, the compiler does not see PowerPlant classes and does not generate information for the symbol database.

The way around this problem is to include a renamed copy of the source file used to create the precompiled headers in your project. These files have names like `PP_ClassHeaders.pch++` or `PP_DebugHeaders.pch++`.

This creates a project-specific precompiled header. By including the source file, CodeWarrior builds the precompiled header from within your project, exposing PowerPlant symbols to the compiler, which generates information for the browser database.

The reason you want to include a renamed copy is to avoid problems if you have multiple projects that use the same precompiled headers. When one project updates the headers, it will be marked as changed for all other projects, which then rebuild the precompiled headers. This defeats the purpose of the precompiled header.

Saving a Default Browser

The browser windows all have various settings that you can modify. You can preserve these modifications as your default settings.

Simply set up a browser window the way that you like. For example, in a Multi-Class Browser Window, set the orientation, the size of each pane, and the size and location of the window. Then choose **Save Default Window** from the Window Menu. The next time you open a multi-class browser window, it will take on the attributes you just set.

You can do the same for any of the browser windows. You must save each window's setup individually, while that window is the active window.

To learn about saving editor windows, see "Saving Editor Window Settings" on page 140.

8

Configuring IDE Options

This chapter discusses the many options available in the CodeWarrior IDE Preferences window. In addition, this chapter discusses the toolbars that appear in various IDE windows and their configuration.

Configuring IDE Overview

You can customize many features of the CodeWarrior IDE through the IDE Preferences window. These global preference settings affect the way the IDE works in all projects.

To view the IDE Preferences window, choose the **Preferences** command from the Edit Menu. The preferences are organized into a series of panels devoted to a particular topic. For example, one panel controls the font and tab settings in the CodeWarrior Editor.

The CodeWarrior IDE's toolbars are another powerful and flexible feature. You can fully customize the toolbars to fit your own working style and thereby make your work more efficient.

The topics in this chapter include:

- Preferences Guided Tour
- Choosing Preferences
- Customizing Toolbars

Preferences Guided Tour

To open the IDE Preferences window, choose the **Preferences** command from the Edit Menu.

The topics in this section are:

- Preference Panels
- Dialog Box Buttons

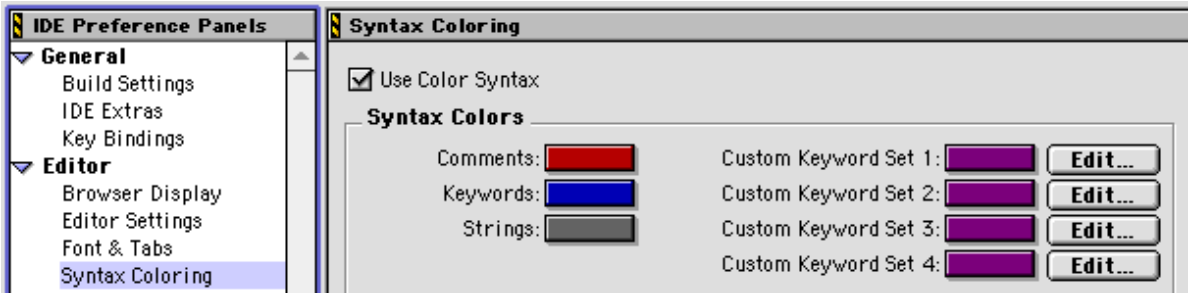
Preference Panels

The IDE Preferences window has a hierarchical list of available panels on its left side. The panel selected from the list appears on its right side. The actual panels available to you may vary, depending upon the CodeWarrior product you are using.

The preference panels affect the IDE as a whole and apply to all projects. To see a panel, select it in the list. You can use the arrow keys or click the name of the panel. Figure 8.1 shows a selected panel in the IDE Preferences window.

Each panel consists of a series of related options that you can set. Once you've made your changes, you can save them, discard them, restore them, or reset them. See "Dialog Box Buttons" for more information.

Figure 8.1 Selecting a preference panel



Dialog Box Buttons

There are several dialog box buttons in the IDE Preferences window that control how a panel's settings are used and applied.

The topics in this section are:

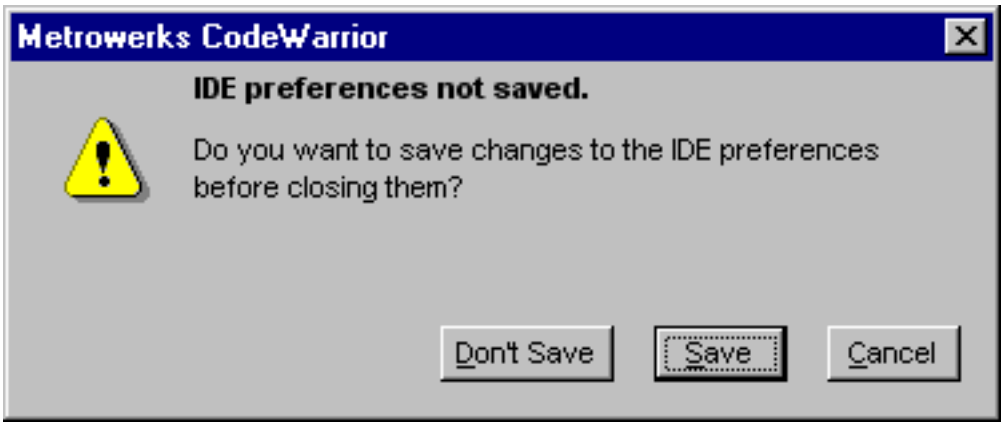
- Discarding Changes

- Factory Settings Button
- Revert Panel Button
- Save Button

Discarding Changes

If you change the settings in the IDE Preferences window and then try to close the window, the CodeWarrior IDE displays the dialog box shown in Figure 8.2. To discard your changes, click the **Don't Save** button. To keep your changes, click the **Save** button. To keep the IDE Preferences window open so that you may continue making changes, click **Cancel**.

Figure 8.2 Preferences Confirmation Dialog Box



Factory Settings Button

The **Factory Settings** button causes the panel to revert to the settings that the CodeWarrior IDE uses as defaults. Settings in other panels are not affected by this button. Only the settings for the current panel are reset.

Revert Panel Button

The **Revert Panel** button allows you to undo the changes that you begin making to a preference panel. Clicking the Revert Panel button causes the currently selected preference panel to reset to its last

saved state. This is useful when you decide against the changes you made.

Save Button

The **Save** button commits any changes you made in any of the panels. After closing the IDE Preferences window, the CodeWarrior IDE will behave according to the preferences that you saved.

Choosing Preferences

This section discusses setting preferences for the IDE as a whole. Here you will learn how to configure preferences that affect the Debugger, Editor, and IDE in general.

To learn how to open and select a particular preference panel, see "Preferences Guided Tour" on page 235.

In this section we discuss each particular preference panel, and what features of the IDE are controlled by it. The panels are:

- General Preferences
- Editor Preferences
- Debugger Preferences

General Preferences

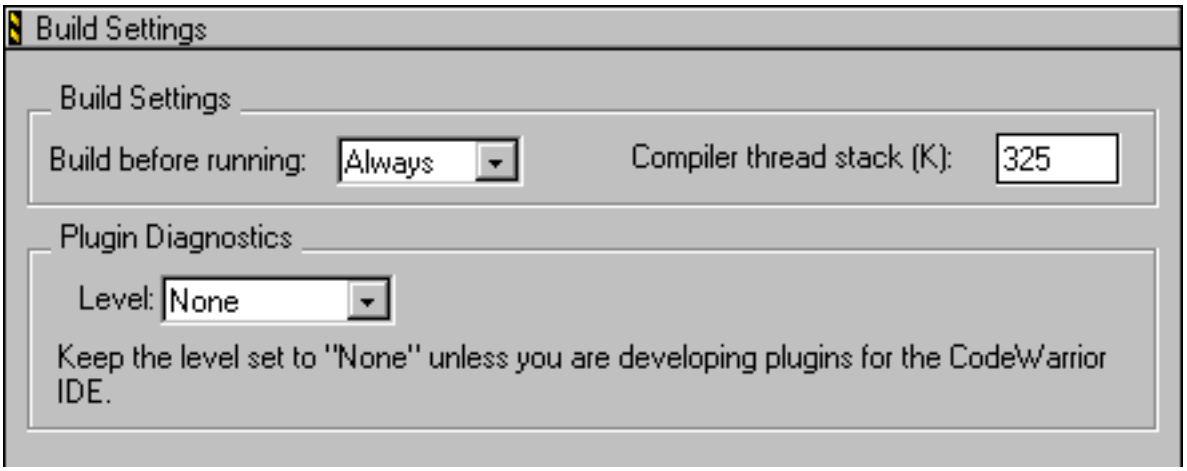
In this section we discuss preference panels that control general IDE features. The General panels include:

- Build Settings Panel
- IDE Extras Panel
- Key Bindings Panel

Build Settings Panel

The Build Settings panel, shown in Figure 8.3 (Windows) and Figure 8.6 (Mac OS), provides project build customizations.

Figure 8.3 Build Settings preference panel (Windows)



Build Settings

These options provide additional customization settings.

Build Before Running

This pop-up menu determines how the CodeWarrior IDE handles project builds. You can choose to always build projects before running them, never build projects before running them, or let the CodeWarrior IDE ask you how to proceed.

Compiler Thread Stack

Use the **Compiler Thread Stack** option, shown in Figure 8.3 (Windows) and Figure 8.6 (Mac OS), to specify the upper limit of stack size allocated by the IDE for compiling and linking thread support.

All builds in CodeWarrior are threaded, with all compilation and linking occurring on a thread separate from the main application thread. This setting allows you to control the size of the stack allocated for this thread.

Normally, you should not change this setting. However, if you have a large or very complex project, you can increase this setting to avoid compiler crashes.

Plugin Diagnostics

This feature allows the IDE to tell you more information about plug-ins that you are developing for CodeWarrior. When you are experiencing problems getting your plug-in to function properly, or you want more information about the properties of installed plug-ins, you should take advantage of this feature.

There are three levels of plug-in diagnostics available:

- **None**—This is the default setting. No plug-in diagnostics take place, and no output is produced.
- **Errors Only**—This setting causes CodeWarrior to report problems that the IDE encounters when loading plug-ins. These problems are displayed in a new text document after the IDE starts up, as shown in Figure 8.4.
- **All Info**—This setting causes CodeWarrior to report detailed information for each plug-in. Problems with loading plug-ins, optional plug-in information, and plug-in properties are reported. This information is displayed in a new text document after the IDE starts up, as shown in Figure 8.5. This text document also includes a complete list of installed plug-ins and their associated preference panels, compilers, and linkers.

In order to use a specific plug-in diagnostic level, select it from the Level pop-up list (Figure 8.6).

NOTE: You must restart CodeWarrior for any changes to the diagnostic level to take effect. When you change a plug-in diagnostic level and save that change, CodeWarrior will display a dialog box reminding you to restart.

After the text file is generated, you have the option of saving the text file. You can also print out the text file so that you can have a convenient error reference when troubleshooting your plug-ins. CodeWarrior provides suggestions for correcting general plug-in errors in the generated text file.

Figure 8.4 Plug-In Diagnostic Errors Only text file

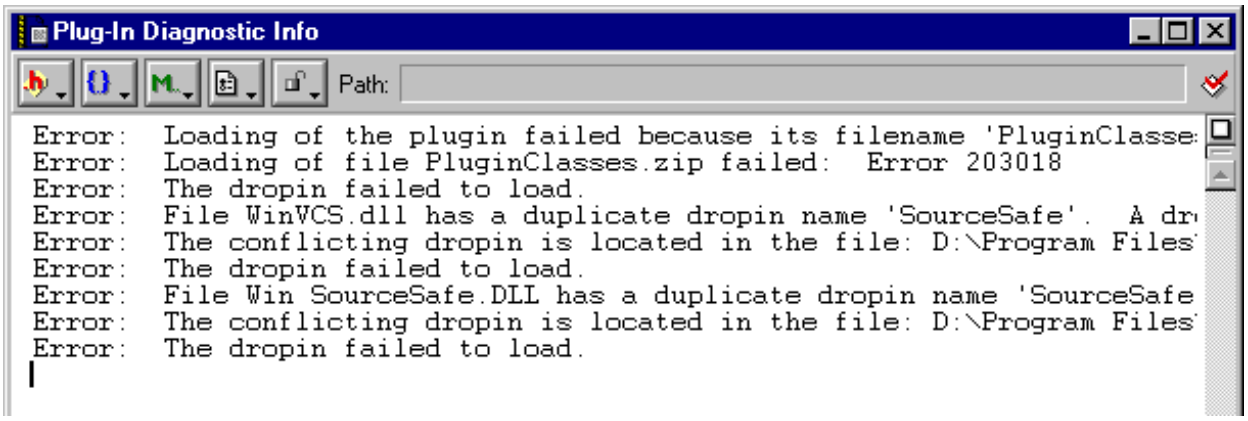
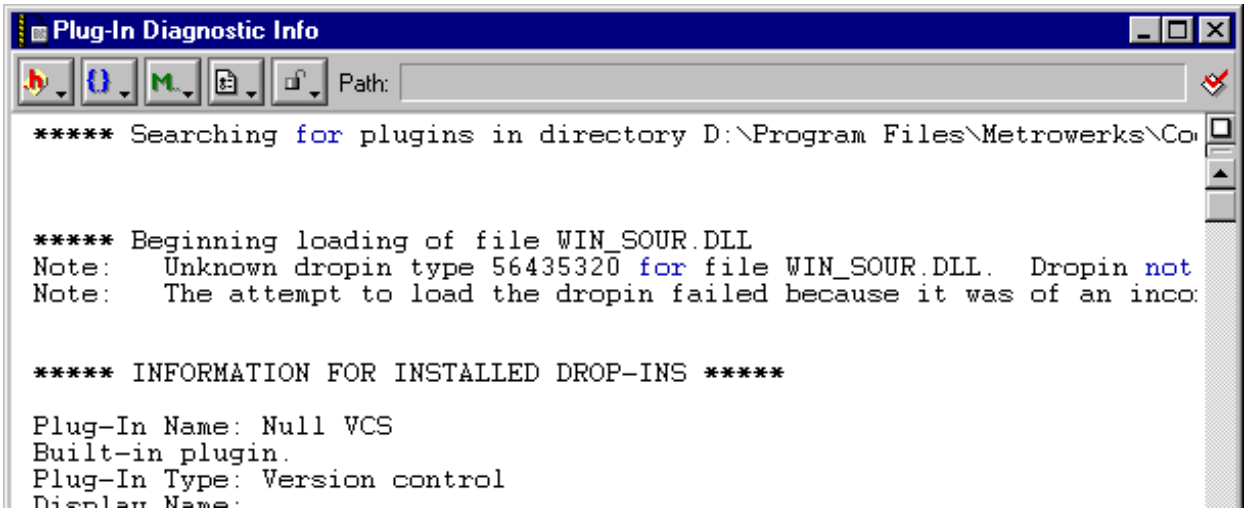

Plug-In Diagnostic Info
Path:
Error: Loading of the plugin failed because its filename 'PluginClasse
Error: Loading of file PluginClasses.zip failed: Error 203018
Error: The dropin failed to load.
Error: File WinVCS.dll has a duplicate dropin name 'SourceSafe'. A dr
Error: The conflicting dropin is located in the file: D:\Program Files
Error: The dropin failed to load.
Error: File Win SourceSafe.DLL has a duplicate dropin name 'SourceSafe
Error: The conflicting dropin is located in the file: D:\Program Files
Error: The dropin failed to load.

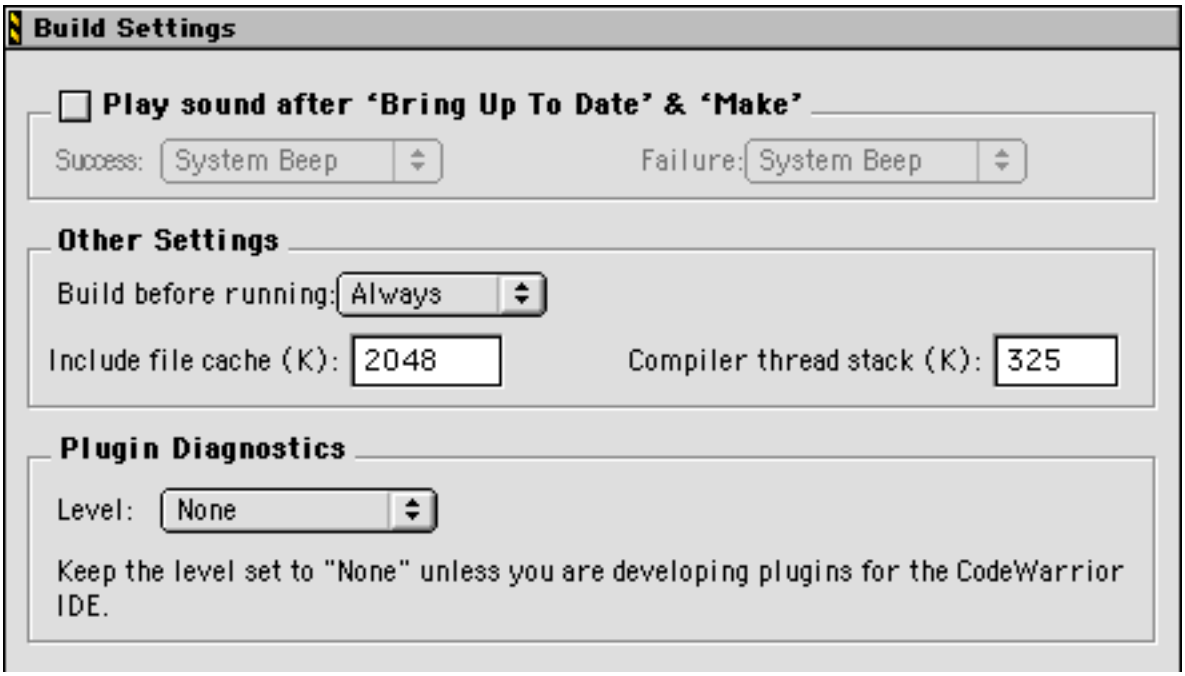
Figure 8.5 Plug-In Diagnostic All Info text file


Plug-In Diagnostic Info
Path:
***** Searching for plugins in directory D:\Program Files\Metrowerks\Co
***** Beginning loading of file WIN_SOUR.DLL
Note: Unknown dropin type 56435320 for file WIN_SOUR.DLL. Dropin not
Note: The attempt to load the dropin failed because it was of an inco
***** INFORMATION FOR INSTALLED DROP-INS *****
Plug-In Name: Null VCS
Built-in plugin.
Plug-In Type: Version control

Include File Cache (Mac OS)

Use the **Include File Cache** option, shown in Figure 8.6, to specify the upper limit on how much memory the IDE should use for caching include files and precompiled headers. If you have a lot of memory and want to use it to speed up builds, increase the number shown in the text box.

Figure 8.6 Build Settings preference panel (Mac OS)


Build Settings
Play sound after 'Bring Up To Date' & 'Make'
Success: System Beep
Failure: System Beep
Other Settings
Build before running: Always
Include file cache (K): 2048
Compiler thread stack (K): 325
Plugin Diagnostics
Level: None
Keep the level set to "None" unless you are developing plugins for the CodeWarrior IDE.

Play Sound After 'Bring Up To Date' & 'Make' (Mac OS)

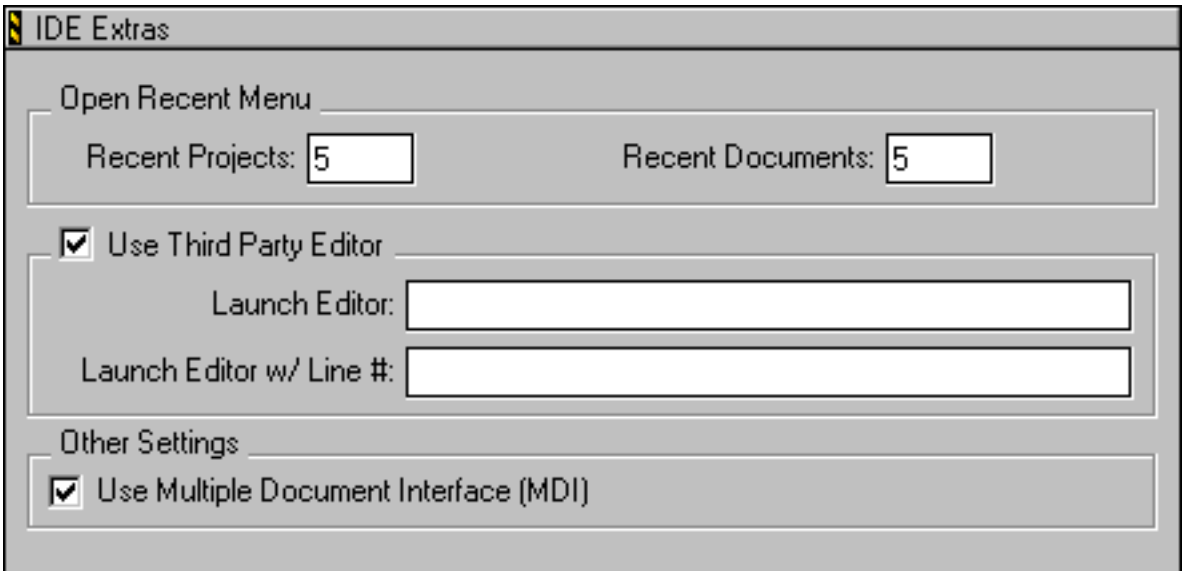
Use the **Play Sound After** option shown in Figure 8.6 to configure the sound that is played when the CodeWarrior IDE finishes a build of your project. You may select different sounds for both successful and unsuccessful build results, using the **Success** and **Failure** pop-up menus.

To add new sounds to this list, you must install additional sounds in the active System Folder on your Mac OS computer. Use the sound control panel on your Mac OS computer to change the sound volume.

IDE Extras Panel

The IDE Extras panel, shown in Figure 8.7, has options to remember previously opened projects and text files and to configure the IDE for use with third-party editors.

Figure 8.7 IDE Extras preference panel (Windows)



Mac OS The IDE Extras preference panel also allows you to configure miscellaneous options such as window zooming, the Scripts Menu (Mac OS), multiprocessor compilation, and on-line reference databases.

Open Recent Menu

These options control the Recent Projects and Recent Documents settings.

Recent Projects

The number entered into this text box is the maximum number of projects that the CodeWarrior IDE will display in the **Open Recent** command on the File Menu.

Recent Documents

The number entered into this text box is the maximum number of files that the CodeWarrior IDE will display in the **Open Recent** Command on the File Menu.

Use Third Party Editor (Windows)

Use this option to control whether the CodeWarrior text editor is used to display text files. If this option is not checked, the CodeWarrior Editor will be the default editor for your text files. When this option is checked, CodeWarrior uses the third-party text editor you specify to display text files.

There are two command lines used for invoking a third-party editor. The first, **Launch Editor**, specifies the text editor that the IDE will launch to display text files. The second command line, **Launch Editor w/ Line #**, specifies a text editor and an initial line of text to jump to upon launch. For example, the IDE invokes this second command line when you double-click on an error message in order to display the line in the text file that caused the error message.

Two variables can be used in the command line string: `%file` and `%line`. When the IDE encounters these variables, it expands `%file` into the full file path and `%line` into the initial line number in the file. For example, if you wanted to use the Emacs text editor to edit text files, then you would type the following command line into the Launch Editor text box:

```
runemacs %file
```

If you also want the text editor to jump to a particular line in the text file, you would type the following command line into the Launch Editor w/ Line # text box:

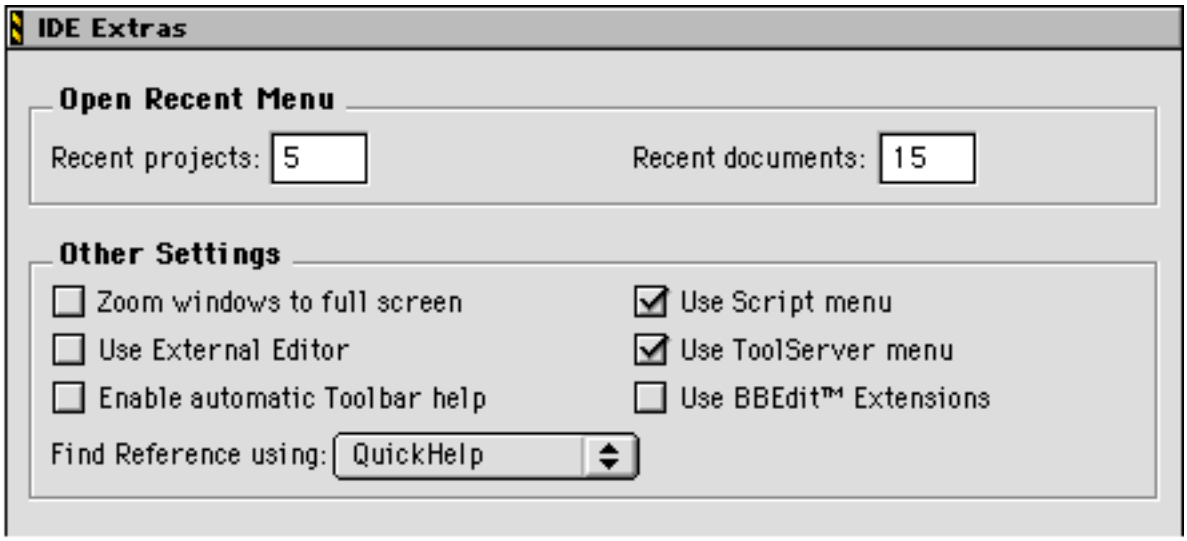
```
runemacs +%line %file
```

For more information on using command lines with a particular text editor, consult the documentation for that editor.

Use Multiple Document Interface (Windows)

Use this option to control which Windows interface style is used by the CodeWarrior IDE. Enable to use MDI (Multiple Document Interface), or disable to use FDI (Floating Document Interface).

Figure 8.8 IDE Extras preference panel (Mac OS)



Zoom Windows To Full Screen (Mac OS)

This option configures the behavior of the zoom box in the upper right-hand corner of all Editor windows. If this option is enabled and the zoom box of an Editor window is clicked, the window will be resized to fill the entire screen. If this option is not checked, the zoom box will cause the window to resize to a standard size.

Use External Editor (Mac OS)

If this option is enabled, the CodeWarrior IDE will send open document '`odoc`' AppleEvents to AppleScript-compatible third-party text editors. To use this feature, create a folder named `(Helper Apps)` in the folder the CodeWarrior IDE application is in (if it doesn't already exist), and create an alias named `External Editor` inside this folder which points to your third-party editor application.

If this option is not checked, the CodeWarrior Editor will be the default editor for your text files.

Note that the third-party editor will not be used for text editing unless the file you are working with has been added to the currently-open Project window.

To learn how to add files to the Project window, refer to "Adding Files" on page 71.

Enable Automatic Toolbar Help (Mac OS)

This option turns on Balloon Help for the toolbar icons. When you move the mouse over an icon and leave it there for a second or two, a balloon will pop up and tell you what command the icon represents.

Use Script Menu (Mac OS)

When this checkbox is clicked, the Scripts Menu will be shown in the CodeWarrior IDE menu bar.

For more information about the Scripts Menu, refer to "Scripts Menu (Mac OS)" on page 413. Additional information about AppleScripts can be found in "CodeWarrior Apple Events Overview" on page 433.

Use ToolServer Menu (Mac OS)

This check box causes the Tools menu (Figure 8.9) to appear in the menu bar. For more information about using this menu, refer to "Tools Menu (Mac OS)" on page 410.

Figure 8.9 ToolServer Menu

Use BBEdit™ Extensions (Mac OS)

When this option is selected, an additional menu bar icon appears in the CodeWarrior IDE menu bar, as shown in Figure 8.9. This icon is useful for accessing BBEdit extensions from within CodeWarrior.

Figure 8.10 BBEdit Extensions Menu Icon

To learn more about the Editor Extensions menu, refer to "To learn more about how to configure the appropriate preferences so this menu becomes visible, refer to "IDE Extras Panel" on page 242." on page 414.

For more information about BBEdit, refer to the documentation that came with the product.

Find Reference Using (Mac OS)

This feature lets you select an online database application to look up references and definitions. QuickHelp-based documents supplied with the CodeWarrior IDE, the Symantec THINK Reference database (not part of the CodeWarrior product), the PalmQuest reference for the PalmPilot, and QuickView-based documents such as the Macintosh Programmer's Toolbox Assistant (MPTA), are supported. CodeWarrior documentation in QuickHelp format can be found on the CodeWarrior Reference CD.

To learn more about on-line reference databases and how to use them, refer to "Online References" on page 155.

Key Bindings Panel

You can customize the keyboard shortcuts used for menu, keyboard, and editor commands in the CodeWarrior IDE. You can at-

tach or "bind" almost any key to any command. This feature allows you to change the keys that activate many commands in the CodeWarrior IDE.

For example, you may wish to change the key equivalent for "Move line up" from the Up-Arrow to Shift-Up-Arrow. You could change this keyboard shortcut by using the key bindings panel.

The Key Bindings preference panel is shown in Figure 8.11 on page 251. You can set the key bindings for menu commands, source code editor actions, and other miscellaneous actions. You can also specify special prefix keys.

NOTE: (Solaris) By default, the Solaris-hosted CodeWarrior IDE uses the same modifier key names as used for Mac OS (Shift, Command, Option, and Control). Likewise, the Key Bindings preference panel uses Mac OS symbols to represent modifier keys. Table A.1 on page 418 shows the default modifier key mappings and the symbols used to represent them. On Solaris computers, modifier keys can be mapped to any key on the keyboard via the Keyboard Preferences dialog box in the Info menu. When reading this manual, you will need to keep the Keyboard Preferences settings in mind. See "Keyboard Preferences Dialog Box" on page 539 for more information on changing the default modifier key mappings.

The topics discussed in this section are:

- Restrictions for choosing key bindings
- What is a prefix key?
- What is the Quote Key prefix?
- Setting the Prefix Key Timeout
- Modifying key bindings
- Exporting key bindings
- Importing key bindings
- Miscellaneous bindings

- Editor bindings

Restrictions for choosing key bindings

When you are customizing key bindings, you need to be aware of some restrictions for keys that can and cannot be used. These restrictions are as follows:

- The Escape and Space keys are always invalid for key bindings.
- Function keys and the Clear key are valid for creating key bindings.
- (Windows) The Return and Tab keys require at least the Control or Shift key. This restriction does not apply for the second key of a two-key sequence.
- (Mac OS) The Return and Tab keys require at least one of the following: Control, Command, or Shift. This restriction does not apply for the second key of a two-key sequence.
- (Mac OS) The Command-period (Command-.) key combination is invalid for any binding.

What is a prefix key?

Prefix keys allow you to create multiple-keystroke command keys, such as those used in the Emacs text editor available on many different computer platforms. For example, the key sequence in Emacs to save a file is Control-X followed by Control-S.

To emulate this Emacs key binding in the CodeWarrior IDE, first set one of the Prefix Keys to be Control-X, and then set the command key for the Save menu command to Control-X Control-S.

You can also adjust the maximum time to wait for a key press after a Prefix Key is entered. To learn how to do this, refer to "Setting the Prefix Key Timeout" on page 250.

What is the Quote Key prefix?

The Quote Key is a special prefix key with a very simple function. It lets you use a simple printing character as a key equivalent (without any modifier key), and still retain the ability to use that character in regular type in the editor window.

In typical use, a key equivalent involves two keys: a modifier key (such as the Control key) combined with an actual printing key. However, you are not required to have a modifier key in CodeWarrior. For example, you can assign the key for the number 1 (with no modifier) to a command.

However, if you do, you can no longer simply type a 1 into your code in the editor. It is interpreted as a command. The Quote Key prefix is the way around this conflict.

In the Key Bindings preference panel, in the prefix section, you can assign any key to be recognized as the Quote Key prefix. Despite the name, it doesn't have to be the key that creates quote symbols in text.

If you have assigned a Quote Key prefix and you then type the Quote Key prefix, CodeWarrior will interpret the next keypress as a keystroke, not as a command. It's that simple.

Returning to the earlier example, assume you have assigned the 1 key to a command. Assume also that you have assigned the tilde key (~) to be your Quote Key prefix. To issue the command, you press the 1 key. To type a 1 into the editor, you would type the tilde key first, then type the 1 key. (To type a tilde, you would press the tilde key twice.)

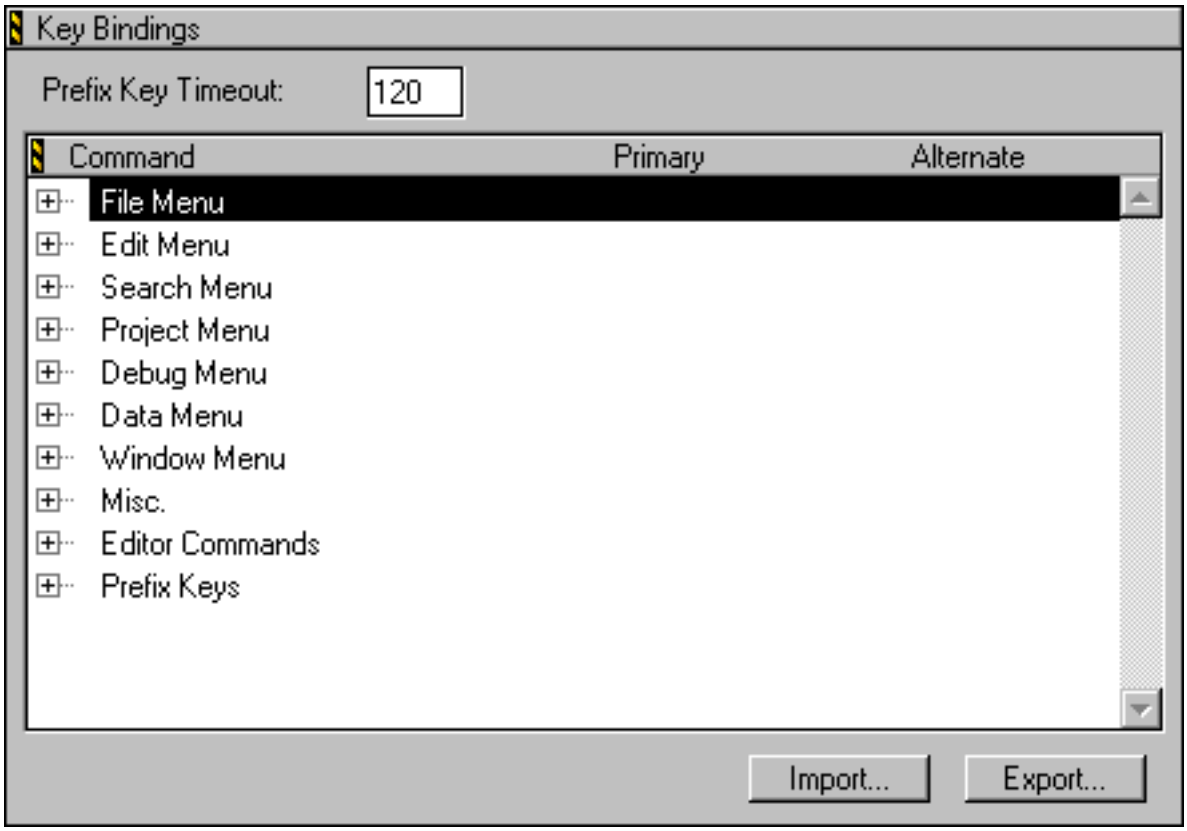
WARNING! The Quote Key only affects the next key or combination of keys that you type. You must use the Quote Key once for each bound key or combination of keys for which you want to type the equivalent character on-screen.

Setting the Prefix Key Timeout

The Prefix Key Timeout field sets the length of time that the IDE waits for the second key after a prefix key is pressed. Larger values mean that the IDE will wait longer for the second key to be pressed.

This value is in "ticks" which are the same as 1/60th of a second (16.67 milliseconds). Legal values are in the range of 1 to 999. The default is 120.

Figure 8.11 Key Bindings preference panel (Windows)



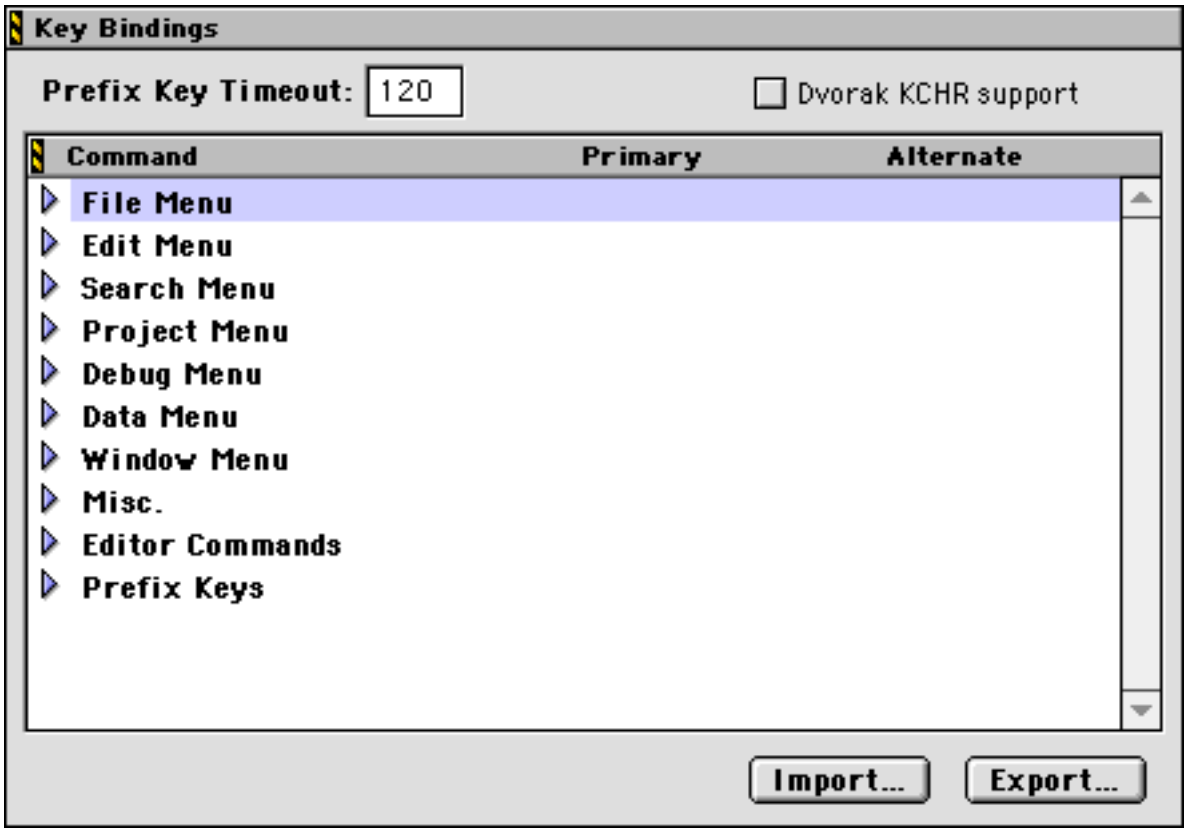
Modifying key bindings

When you select the Key Bindings item from the list of IDE preference panels, you will see the Key Bindings panel shown in Figure 8.11 (Windows) or Figure 8.12 (Mac OS).

To change the key that will activate a given IDE operation, you must first select the operation you wish to modify. To do this, click the tree control (Windows) or disclosure triangle (Mac OS) next to the Command category to show the individual operations.

Mac OS If you are using a Dvorak keyboard and experiencing problems with the Command key being recognized, enable the **Dvorak KCHR Support** option.

Figure 8.12 Key Bindings preference panel (Mac OS)



To change a key binding, double-click it or select it and press the Return/Enter key. The dialog box shown in Figure 8.13 appears.

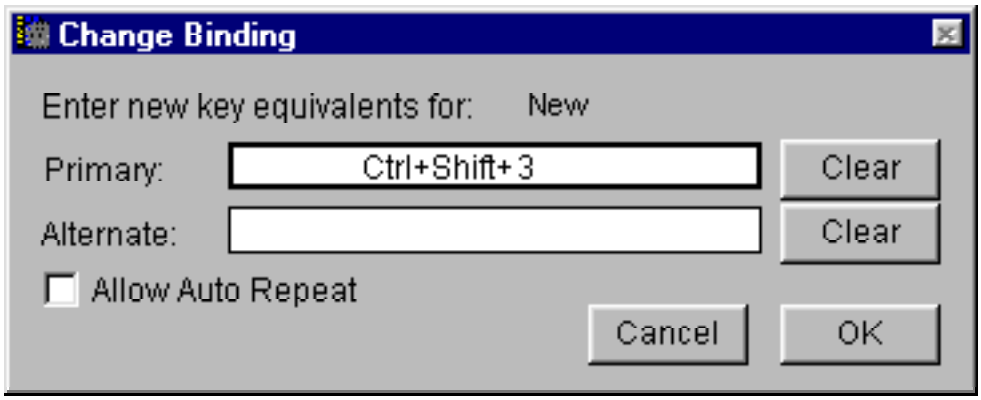
Once you see this dialog, press the key combination you want to use for the command. For example, if you want to make the command key Control-8, you would press the Control key and the 8 key at the same time.

You may also specify an alternate key that can be used for the command. To create an additional command key, click on the Alternate rectangle so that it is selected. Then, hold down the keys that you want to use for the command. If you specify an alternate command key, either key can be used to execute the command.

The only exception to the alternate key feature is prefix keys. Note that you cannot specify an alternate key for any prefix key.

If you wish to clear the Primary or Alternate command keys, click the **Clear** button to the right of the appropriate field.

Figure 8.13 Change Key Binding Dialog Box



Use the **Allow Auto Repeat** option to make a command key repeat while you continue to press the key combination. A good key binding to use here as an example is the **Find Next** command. If you select **Allow Auto Repeat** for the Find Next key binding, then you can just hold down the key combination while you watch the search engine find all the text matching your search criteria in the currently-open file. This is a quick way to jump through a file, finding all the patterns you are looking for and showing them quickly. If you did not configure this feature to **Allow Auto Repeat**, then you would have to release the keys and press them again every time.

When you are finished changing key bindings, click the **OK** button to dismiss the dialog box, saving your changes. If you do not wish to save your changes, click the **Cancel** button.

Exporting key bindings

If you wish to save your key bindings in a file so that you can later import them into the IDE at another time, you use the **Export** button. When you click this button, a standard file dialog box appears. Navigate to the place on your hard disk where you want to save the key bindings file, and click **Save**.

Importing key bindings

If you wish to import saved key bindings from a previously-exported file, you use the **Import** button. Use the standard file dialog to locate and open the key bindings file.

Miscellaneous bindings

Table 8.1 lists the default key equivalents for some miscellaneous IDE commands, and describes what each command does.

Table 8.1 Miscellaneous key bindings

Command	Windows Default Key Binding	Mac OS Default Key Binding	Solaris Default Key Binding	Description
Go to header/source file	Ctrl-`	Command-Tab	Command-Tab	Switches between a source and header file of the same name. See "Opening a Related File" on page 153.
Go to previous error message	Shift-F4	Option-Command Up Arrow	Option-Command Up Arrow	Moves to the previous error message in the Message window. See "Stepping Through Messages" on page 342.
Go to next error message	F4	Option-Command Down Arrow	Option-Command Down Arrow	Moves to the next error message in the Message window. See "Stepping Through Messages" on page 342.

Editor bindings

Table 8.2 lists the default key equivalents for each command in the editor. To learn more about text navigation features of the Editor, refer to "Navigating the Text" on page 150.

Table 8.2 Editor key bindings

Command	Windows Default Key Binding	Mac OS Default Key Binding	Solaris Default Key Binding	Description
Move character left	Left Arrow	Left Arrow	Left Arrow	Moves the text insertion point one character to the left.
Move character right	Right Arrow	Right Arrow	Right Arrow	Moves the text insertion point one character to the right.
Move word left	Ctrl-Left Arrow	Option-Left Arrow	Option-Left Arrow	Moves the text insertion point one word to the left.
Move word right	Ctrl-Right Arrow	Option-Right Arrow	Option-Right Arrow	Moves the text insertion point one word to the right.
Move sub-word left	Alt-Left Arrow	Control-Left Arrow	Control-Left Arrow	Moves the text insertion point to the left of the next capital letter or start of a word on the left.
Move sub-word right	Alt-Right Arrow	Control-Right Arrow	Control-Right Arrow	Moves the text insertion point to the right of the next capital letter or start of a word on the right.
Move to start of line	Home	Command-Left Arrow	Command-Left Arrow	Moves the text insertion point to the start of the current line.

Command	Windows Default Key Binding	Mac OS Default Key Binding	Solaris Default Key Binding	Description
Move to end of line	End	Command-Right Arrow	Command-Right Arrow	Moves the text insertion point to the end of the current line.
Move line up	Up Arrow	Up Arrow	Up Arrow	Moves the text insertion point one line up.
Move line down	Down Arrow	Down Arrow	Down Arrow	Moves the text insertion point one line down.
Move to top of page	Page Up	Option-Up Arrow	Option-Up Arrow	Moves the text insertion point to the top of the current view.
Move to bottom of page	Page Down	Option-Down Arrow	Option-Down Arrow	Moves the text insertion point to the bottom of the current view.
Move to top of file	Ctrl-Home	Command-Up Arrow	Command-Up Arrow	Moves the text insertion point to the top of the current file.
Move to bottom of file	Ctrl-End	Command-Down Arrow	Command-Down Arrow	Moves the text insertion point to the bottom of the current file.
Delete character left	Backspace	Delete	Backspace	Erases the last character typed.
Delete character right	Delete	Del (⌦ key)	Del	Deletes the character immediately to the right of the text insertion point.

Command	Windows Default Key Binding	Mac OS Default Key Binding	Solaris Default Key Binding	Description
Delete to end of file	Ctrl-Backspace	Command-Delete	Command-Backspace	Deletes the characters from the text insertion point to the end of the file.
Character select left	Shift-Left Arrow	Shift-Left Arrow	Shift-Left Arrow	Adds the character to the left of the text insertion point to the current selection.
Character select right	Shift-Right Arrow	Shift-Right Arrow	Shift-Right Arrow	Adds the character to the right of the text insertion point to the current selection.
Select word left	Ctrl-Shift-Left Arrow	Option-Shift-Left Arrow	Option-Shift-Left Arrow	Selects the full word to the left of the text insertion point.
Select word right	Ctrl-Shift-Right Arrow	Option-Shift-Right Arrow	Option-Shift-Right Arrow	Selects the full word to the right of the text insertion point.
Select sub-word left	Alt-Shift-Left Arrow	Control-Shift-Left Arrow	Control-Shift-Left Arrow	Selects the partial word to the left of the text insertion point, up to the first capital letter or white space, whichever comes first.
Select sub-word right	Alt-Shift-Right Arrow	Control-Shift-Right Arrow	Control-Shift-Right Arrow	Selects the partial word to the right of the text insertion point, up to the first capital letter or white space, whichever comes first.

Command	Windows Default Key Binding	Mac OS Default Key Binding	Solaris Default Key Binding	Description
Select line up	Shift-Up Arrow	Shift-Up Arrow	Shift-Up Arrow	Selects all text from the current location of the text insertion point to a position one line directly above.
Select line down	Shift-Down Arrow	Shift-Down Arrow	Shift-Down Arrow	Selects all text from the current location of the text insertion point to a position one line directly below.
Select to start of line	Shift-Home	Shift-Command-Left Arrow	Shift-Command-Left Arrow	Selects all text left of the text insertion point through the beginning of the current line.
Select to end of line	Shift-End	Shift-Command-Right Arrow	Shift-Command-Right Arrow	Selects all text right of the text insertion point through the end of the current line.
Select to start of page	Shift-Page Up	Option-Shift-Up Arrow	Option-Shift-Up Arrow	Selects all text left of the text insertion point up to the top of the current view of the file.
Select to end of page	Shift-Page Down	Option-Shift-Down Arrow	Option-Shift-Down Arrow	Selects all text right of the text insertion point down to the bottom of the current view of the file.

Command	Windows Default Key Binding	Mac OS Default Key Binding	Solaris Default Key Binding	Description
Select to start of file	Ctrl-Shift-Home	Shift-Command-Up Arrow	Shift-Command-Up Arrow	Selects all text left of the text insertion point up to the top of the file.
Select to end of file	Ctrl-Shift-End	Shift-Command-Down Arrow	Shift-Command-Down Arrow	Selects all text right of the text insertion point down to the bottom of the file.
Scroll line up	Ctrl-Up Arrow	Control-Up Arrow	Control-Up Arrow	Scrolls the view up one line.
Scroll line down	Ctrl-Down Arrow	Control-Down Arrow	Control-Down Arrow	Scrolls the view down one line.
Scroll page up	Page Up	Page Up	Page Up	Scrolls the view up one page.
Scroll page down	Page Down	Page Down	Page Down	Scrolls the view down one page.
Scroll to top of file	Home	Home	Home	Scrolls the view up to the top of the file.
Scroll to end of file	End	End	End	Scrolls the view down to the bottom of the file.
Find symbols with prefix	Ctrl-\	Control-\	Control-\	Enters the first browser item that is matched by first part of the selected or just-entered text.

Command	Windows Default Key Binding	Mac OS Default Key Binding	Solaris Default Key Binding	Description
Find symbols with substring	Ctrl-Shift-\	Control-Shift-\	Control-Shift-\	Enters the first browser item that is matched by any part of the selected or just-entered text.
Get next symbol	Ctrl-.	Control-.	Control-.	Enters the next matching browser item.
Get previous symbol	Ctrl-,	Control-,	Control-,	Enters the previous matching browser item.

Editor Preferences

In this section we discuss preference panels that control Editor features. The Editor panels include:

- Browser Display
- Editor Settings
- Font & Tabs
- Syntax Coloring

Browser Display

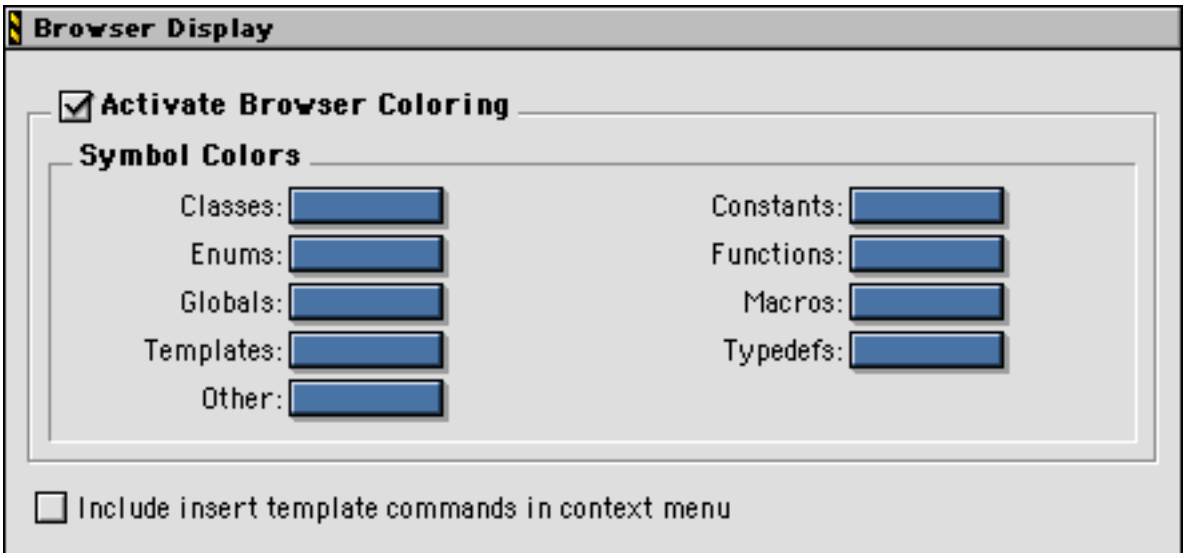
The Browser Display preferences are shown in Figure 8.14.

The Browser can export its lists of symbols and their types to the Editor. This enables the Editor to use different colors for displaying various types of symbols. Choose **Activate Browser Coloring** to use this feature. When active, the color choice for each symbol type will be displayed in the Editor window and the Browser window. Click on a color sample to change its color.

Enable **Include insert template commands in context menu** to use the context pop-up menus to insert templates into your source code.

By default, this option is disabled. When this option is enabled, you will see an additional **Insert Template** command in your context pop-up menus, as shown in Figure 7.4 on page 209.

Figure 8.14 Browser Display options

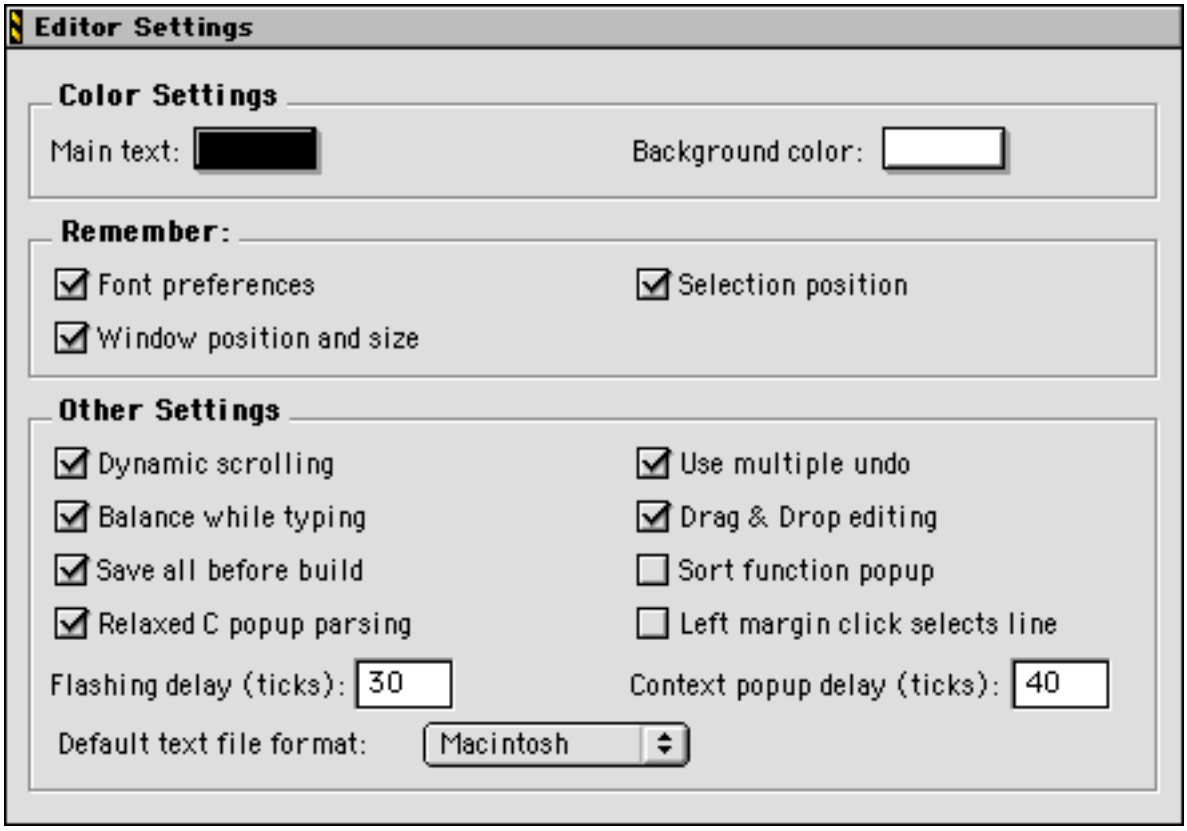


Editor Settings

This section tells you how to configure the Editor's behavior to make your text editing chores easier. The Editor preferences panel is shown in Figure 8.15.

This preference panel has three areas of options: Color Settings, Remember, and Other Settings. Color Settings controls the main text color (non-syntax) and the background color in the editor and browser windows. The Remember options determine the Editor window settings that are saved from one programming session to the next. Other Settings (such as Dynamic Scrolling and Balance While Typing) control how the Editor works.

Figure 8.15 Editor Settings preference panel



Color Settings

The following options control the color settings for the Editor.

Main Text Color This option configures the color of any text not colored by the Browser Display, Syntax Coloring, or Custom Keywords color sets. Click the color sample to change the color using the operating system's standard color selection window.

Background Color Click the color sample to change the background color of the Editor and Browser windows using the operating system's standard color selection window.

Remember

These options control which settings the Editor remembers between programming sessions.

Font Preferences You can configure the font information for an individual file if you use this option. Otherwise, all files inherit the default font settings from the CodeWarrior IDE.

Window Position and Size This option saves the window position and size so files open in the same location on the screen each time. This feature requires that your Editor files be writable. To learn more about writable files, refer to "Using Source Code Control with Files" on page 359.

Selection Position This option tells the CodeWarrior IDE to remember what text was scrolled into view, and the location of the insertion point or selection. Turn this option off if you always want the editor to go to the top of the file when it opens a window. This feature requires that your files be writable. To learn more about writable files, refer to "Using Source Code Control with Files" on page 359.

Other Settings

These options control other behaviors that the Editor remembers between programming sessions.

Dynamic Scrolling (Mac OS) When this option is enabled, the text in Editor windows visually scrolls while you drag the scroll box in the scrollbar. To temporarily disable this option, hold down the Option key while dragging the scroll box.

Balance While Typing When the Balance While Typing option is enabled, the CodeWarrior IDE checks for balanced parentheses, brackets, and braces as you type. Each time you type a right parenthesis, bracket, or brace, the Editor attempts to locate the matching left counterpart. If the counterpart is found, the Editor brings it into view, highlights it for a specified length of time called the Flashing Delay, and returns to where you were typing. If the counterpart is not found, the Editor beeps. By default, the Balance While Typing option is on. To learn more about Flashing Delay, refer to "Flashing Delay" on page 264.

TIP: If you want to check for balanced punctuation without highlighting it, set the Flashing Delay to 0.

Save All Before Build Enable this option if you want to save all text documents automatically before a **Preprocess**, **Precompile**, **Compile**, **Disassemble**, **Bring Up To Date**, **Make**, or **Run** command is executed.

Relaxed C Popup Parsing Enable this option if you use K&R style coding conventions in your source code to allow the CodeWarrior IDE to recognize and display function names in the Routine Pop-up menu. Disable this option if you use non-standard macros that can interfere with K&R styled code.

NOTE: Some macro functions will not be recognized when this option is enabled. If you encounter problems with viewing routine names, disable this option and try again.

Use Multiple Undo When active, you can undo and redo multiple actions. When this option is turned off, you can only undo or redo the last action that you performed. See "Redo, Multiple Undo, and Multiple Redo" on page 379 for more information.

Drag & Drop Editing This option enables Drag and Drop text editing support in the Editor. To learn more about Drag & Drop Editor features, refer to "Moving Text (Drag and Drop)" on page 146.

Sort Function Popup Enable this option if you want the Routine Pop-Up Menu in the Editor window to be sorted by default. To learn more about this feature, refer to "Routine Pop-Up Menu" on page 132.

Left Margin Click Selects Line When this option is enabled, moving the mouse pointer to the left edge of an editor window changes the mouse pointer into a right-pointing arrow. Clicking the window when the mouse pointer faces right selects the line at the mouse pointer. Clicking and dragging the mouse when the mouse pointer faces right selects more than one line. When this option is off, the mouse pointer always faces left and cannot select an entire line with a click.

Flashing Delay The Flashing Delay is the amount of time the CodeWarrior Editor displays and highlights an item. It is measured in 60ths of a second. This option is for balancing punctuation. To

learn more about balancing punctuation, refer to "Balancing Punctuation" on page 147 and "Balance While Typing" on page 263.

WARNING! If you enter 0 (zero) for the time delay, you disable flashing entirely.

Context Popup Delay Context Popup Delay determines how long the mouse button must be held down before the browser's Context Pop-Up Menu appears. The range of acceptable values is 0-240. Each interval represents 1/60th of a second (16.67 milliseconds). To learn more about this browser feature, refer to "Context Pop-Up Menu" on page 208 and "Using the Context Pop-Up Menu" on page 227.

WARNING! If you enter 0 (zero) for the time delay, you disable the pop-up menu entirely.

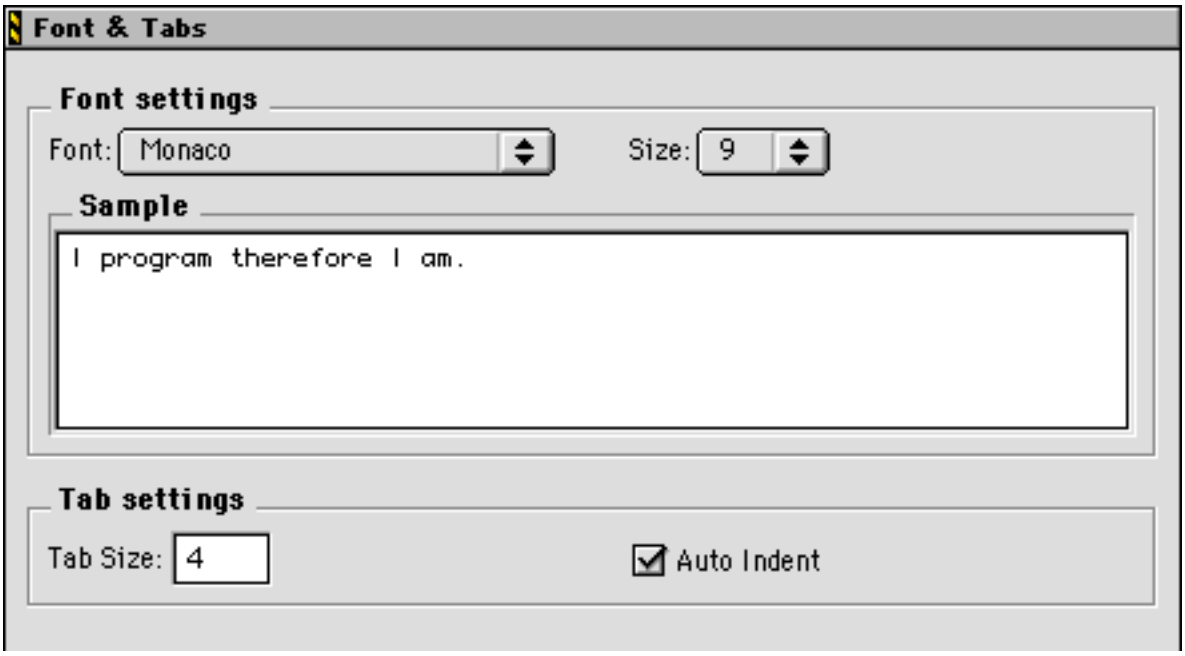
Default Text File Format The Default Text File Format pop-up menu sets the end-of-line conventions that the CodeWarrior IDE uses to create new files. You can choose from three platform text formats: DOS, Macintosh, and UNIX. To learn about saving text files under a different text format, see "Options Pop-Up Menu" on page 134.

Font & Tabs

To change the settings for Font and Tabs you use the preference panel shown in Figure 8.16. This preference panel sets the font and tab information for the active Editor window. If no Editor window is open, this preference panel applies to the CodeWarrior IDE defaults.

To change the font and tab settings for a file, make it the active Editor window, then open the Font & Tabs preference panel and make your changes. Every time you open the file, the Editor remembers the font and tab preferences that you set.

Figure 8.16 Font & Tabs preference panel



As long as you have write permissions on the file (i.e. the file is not Read-Only), your changes will be remembered. To learn more about writable files, refer to "Using Source Code Control with Files" on page 359.

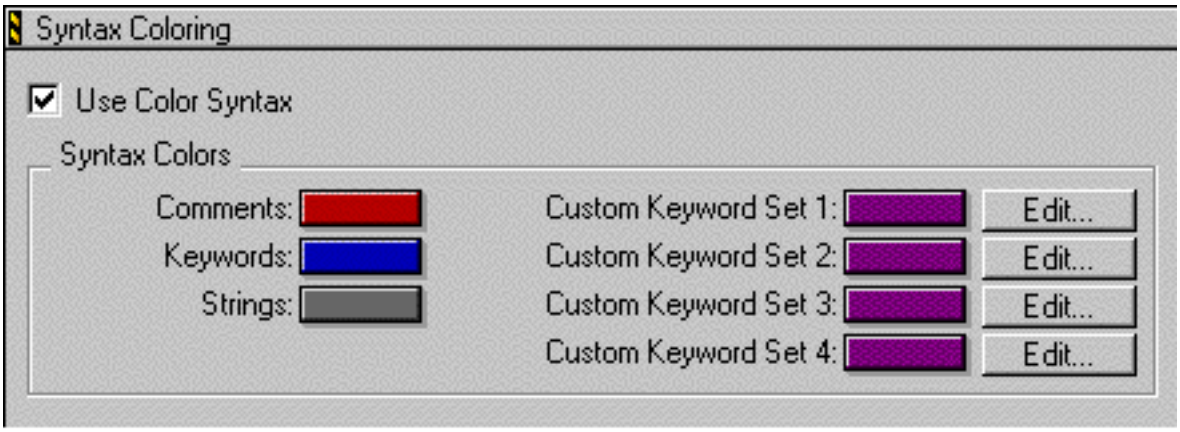
Tab Size is the number of spaces the CodeWarrior IDE inserts in Editor text files when you press the Tab key.

Choose **Auto Indent** if you want the editor to automatically indent text on a new line to match up with the text on the previous line.

Syntax Coloring

The Syntax Coloring preferences panel provides four Custom Keyword Set settings you can use to highlight lists of custom keywords. The list can contain routine names, type names, or anything else you want to have stand out in your Editor windows.

Figure 8.17 Syntax Coloring preference panel



To turn on Syntax Coloring, enable the **Use Color Syntax** checkbox in the top left corner of the preferences panel, as shown in Figure 8.17.

Table 8.3 describes each element of text that the CodeWarrior Editor displays in color.

Table 8.3 Syntax coloring highlights

Element	Description
Main Text	Anything that's not a comment, keyword, or custom keyword, such as literal values, variable names, routine names, and type names.
Comments	Code comments. In Java, C or C++, a comment is text enclosed by `/*` and `*/` or text from `//` to the end of the line. In Pascal, a comment is text enclosed by `{` and `}` or `(*` and `*)`.

Element	Description
Keywords	The language's keywords. It does not include any macros, types, or variables that you or the system interface files define.
Custom Keywords	Any keyword listed in the Custom Keyword List. This list is useful for macros, types, and other names that you want to highlight.

Changing syntax highlighting colors

The CodeWarrior IDE can use different colors for each type of text. To change these colors, click on the color sample beside the name. The CodeWarrior IDE displays the dialog box you use to select a color. The next time you view a text file, the CodeWarrior IDE uses the new color.

Controlling syntax highlighting within a window

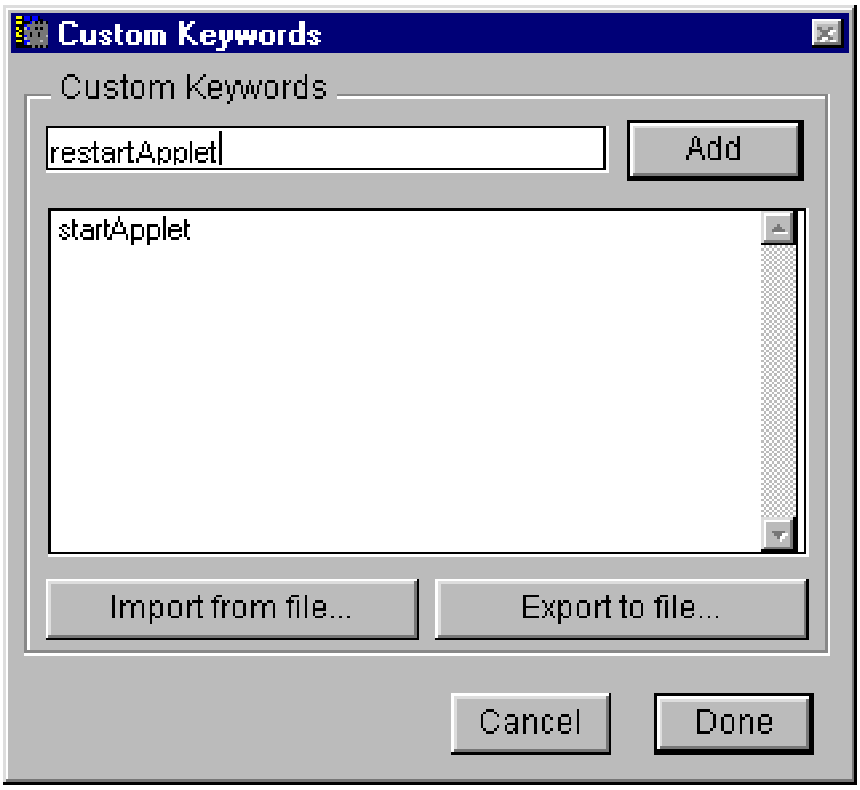
Use the **Syntax Coloring** item in the Options Pop-Up Menu to turn syntax coloring on or off as you view a particular file. For more information, see "Options Pop-Up Menu" on page 134.

Using color for custom keywords

Use the Custom Keywords dialog box to choose additional words to display in color. These words can be macros, types, or other names that you want to highlight. These keywords are global to the CodeWarrior IDE and will apply to every project.

Click the **Edit** button to the right of the Custom Keyword Set you want to modify. The CodeWarrior IDE displays the Custom Keywords dialog box, shown in Figure 8.18.

Figure 8.18 Custom Keywords dialog box



Type a keyword in the Add field, then click **Add**. The CodeWarrior IDE adds the keyword to the Custom Keywords List. You can add as many keywords as you want. However, due to the way that the IDE handles these keywords, you may not be able to add keywords to a large Custom Keywords List. If the IDE is unable to add a keyword to the list, it will display a dialog box informing you that adding the keyword was unsuccessful.

To delete a keyword, select the keyword and then press Backspace/Delete. The CodeWarrior IDE removes the keyword from the Custom Keywords List.

Once you've finished entering custom keywords, click **Done**. The dialog box disappears. When you next view a source file, all the custom keywords you entered are colored.

TIP: You can also set target-specific colors for custom keywords. To learn more about this, see "Custom Keywords" on page 312.

Importing or exporting custom keywords

To retrieve or save an entire group of keywords, use the **Import from File** and **Export to File** buttons.

Click the **Edit** button to the left of the appropriate Custom Keyword Set (shown in Figure 8.17 on page 267).

Choose the **Import** or **Export** button. Complete the standard dialog box that appears. If you are importing custom keywords, use the dialog box to find and open the custom keywords file. If you are exporting custom keywords, use the dialog box to save your custom keywords file to your hard drive.

The CodeWarrior IDE adds or subtracts the custom keywords from the selected file to or from your Custom Keyword Set.

When you're finished, click **Done**. The dialog box disappears. When you next view a source file, all the custom keywords you entered will be highlighted as you designated.

Debugger Preferences

In this section we discuss preference panels that control Debugger features. The Debugger panels include:

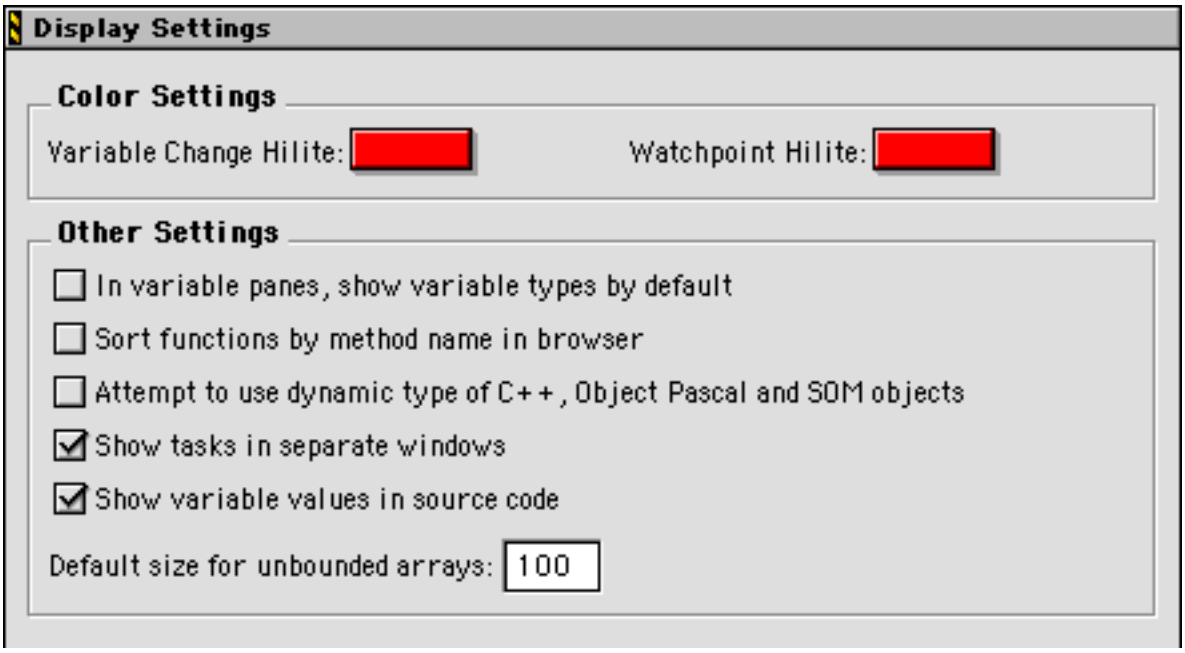
- Display Settings Panel
- Windowing Panel
- Global Settings Panel
- MetroNub Settings Panel (Mac OS)
- Java Settings Panel (Windows)
- x86 Settings Panel (Mac OS)

Display Settings Panel

You can configure the integrated debugger to conform to the way you work. The Debugger Display preferences panel is shown in Figure 8.19.

For more information on the commands contained in the Display Settings preference panel, see *Debugger User Guide*,.

Figure 8.19 Debugger Display Settings preference panel


Display Settings
Color Settings
Variable Change Hilite:
Watchpoint Hilite:
Other Settings
In variable panes, show variable types by default
Sort functions by method name in browser
Attempt to use dynamic type of C++, Object Pascal and SOM objects
Show tasks in separate windows
Show variable values in source code
Default size for unbounded arrays:
100

Variable Change Hilite

Allows you to set the color that the debugger uses to identify a changed variable.

Watchpoint Hilite

Allows you to set the color that the debugger uses to identify a watchpoint.

In variable panes, show variable types by default

Shows variable types when a new variable window is opened.

Sort functions by method name in browser

Changes the way C++, Object Pascal, and Java functions are sorted in the browser window's function pane.

Attempt to use dynamic type of C++, Object Pascal and SOM objects

Displays the runtime type of C++ or Object Pascal objects; deselecting this preference displays an object's static type only.

Show tasks in separate windows

Allows you to toggle between two ways of displaying tasks.

Show variable values in source code

Enable this option to automatically display variable values in your source code. When disabled, the variable values do not appear.

Default size for unbounded arrays

Specifies the array size to use when no size information is available.

Windowing Panel

The Windowing preference panel, shown in Figure 8.20 (Windows) and Figure 8.21 (Mac OS), allows you to configure window behavior when you start a debugging session.

When Debugging Starts

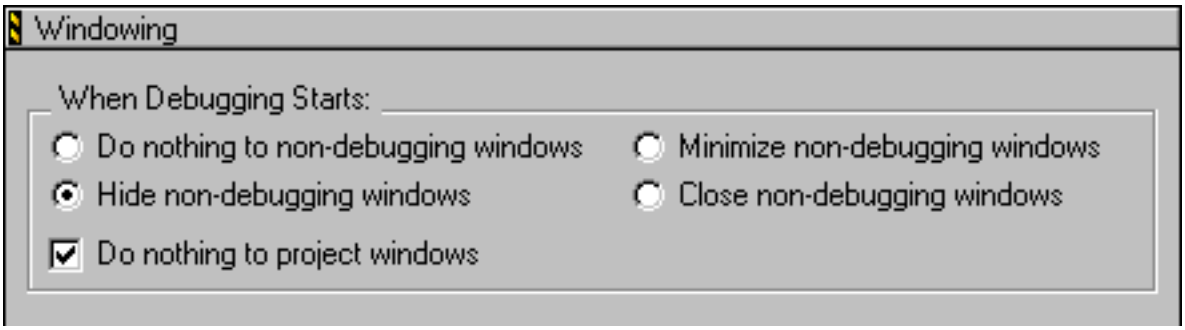
The following settings determine what the IDE does with non-debugging windows when a debugging session begins.

Do nothing to non-debugging windows Enable this option so that non-debugging windows are not closed or hidden when debugging begins. Since projects are used to initiate debugging, this setting is useful in situations where multiple targets or multiple projects are being debugged.

Hide non-debugging windows Enable this option to hide, but not close, non-debugging windows. Use the Window Menu to access the non-debugging windows when they have been hidden. In addition, double-clicking files in the project window and perform-

ing symbol lookups will show the hidden windows. At the end of the debugging session, windows that were hidden will be made visible.

Figure 8.20 Windowing preference panel (Windows)



Minimize non-debugging windows (Windows) Enable this option to minimize non-debugging windows. Use the Window Menu to access the non-debugging windows when they have been minimized. In addition, double-clicking files in the project window and performing symbol lookups will expand the minimized windows. At the end of the debugging session, windows that were minimized will be expanded. Note that this option is available only when CodeWarrior is in MDI mode. For more information, see "IDE Extras Panel" on page 242 and "Use Multiple Document Interface (Windows)" on page 244.

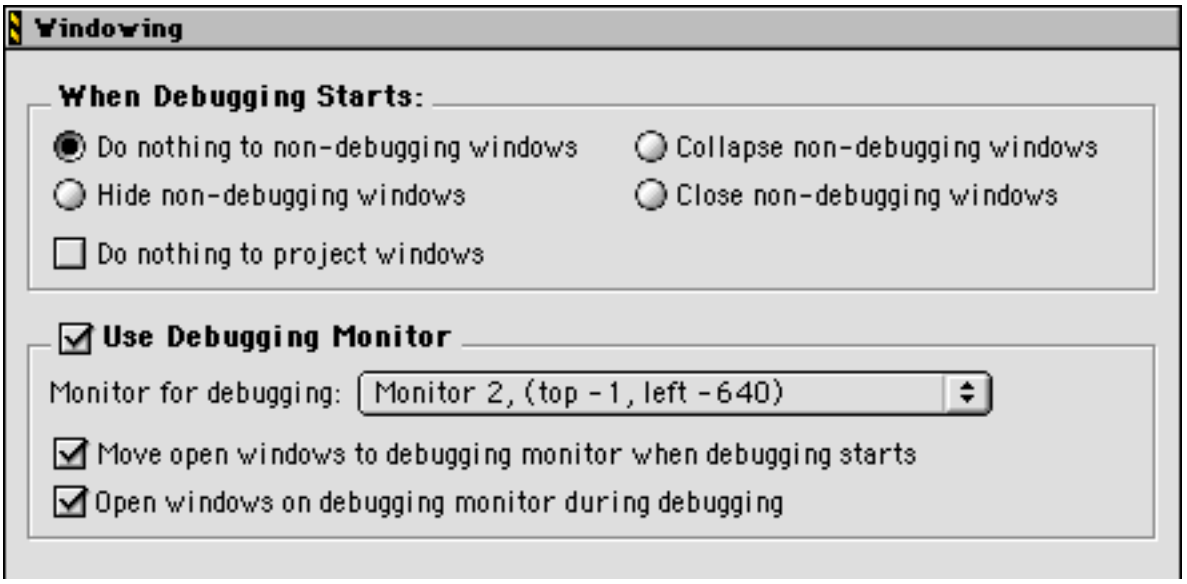
Collapse non-debugging windows (Mac OS) Enable this option to collapse all non-debugging windows. Use the Window Menu to access the non-debugging windows when they have been collapsed. In addition, double-clicking files in the project window and performing symbol lookups will expand the collapsed windows. At the end of the debugging session, windows that were collapsed will be expanded.

Close non-debugging windows Enable this option to close all non-debugging windows, except for the project window of the project being debugged. At the end of the debugging session, windows that were closed will be re-opened.

Multi-Monitor Debugging (Mac OS)

The following options, shown in Figure 8.21, appear only when you have more than one monitor connected to your computer. These options help you manage debugging windows across your multiple-monitor setup. Enable the **Use Debugging Monitor** checkbox to take advantage of CodeWarrior's multiple-monitor debugging features.

Figure 8.21 Windowing preference panel (Mac OS)


Windowing
When Debugging Starts:
Do nothing to non-debugging windows
Collapse non-debugging windows
Hide non-debugging windows
Close non-debugging windows
Do nothing to project windows
Use Debugging Monitor
Monitor for debugging:
Monitor 2, (top -1, left -640)
Move open windows to debugging monitor when debugging starts
Open windows on debugging monitor during debugging

Monitor for debugging Use this pop-up menu to select the monitor you wish to use during debugging sessions. The coordinates in parentheses identify the selected monitor in QuickDraw space.

Move open windows to debugging monitor when debugging starts When you enable this option, all open windows are moved to the selected debugging monitor when you begin a debugging session. At the end of the debugging session, windows that were moved will be restored to their original positions.

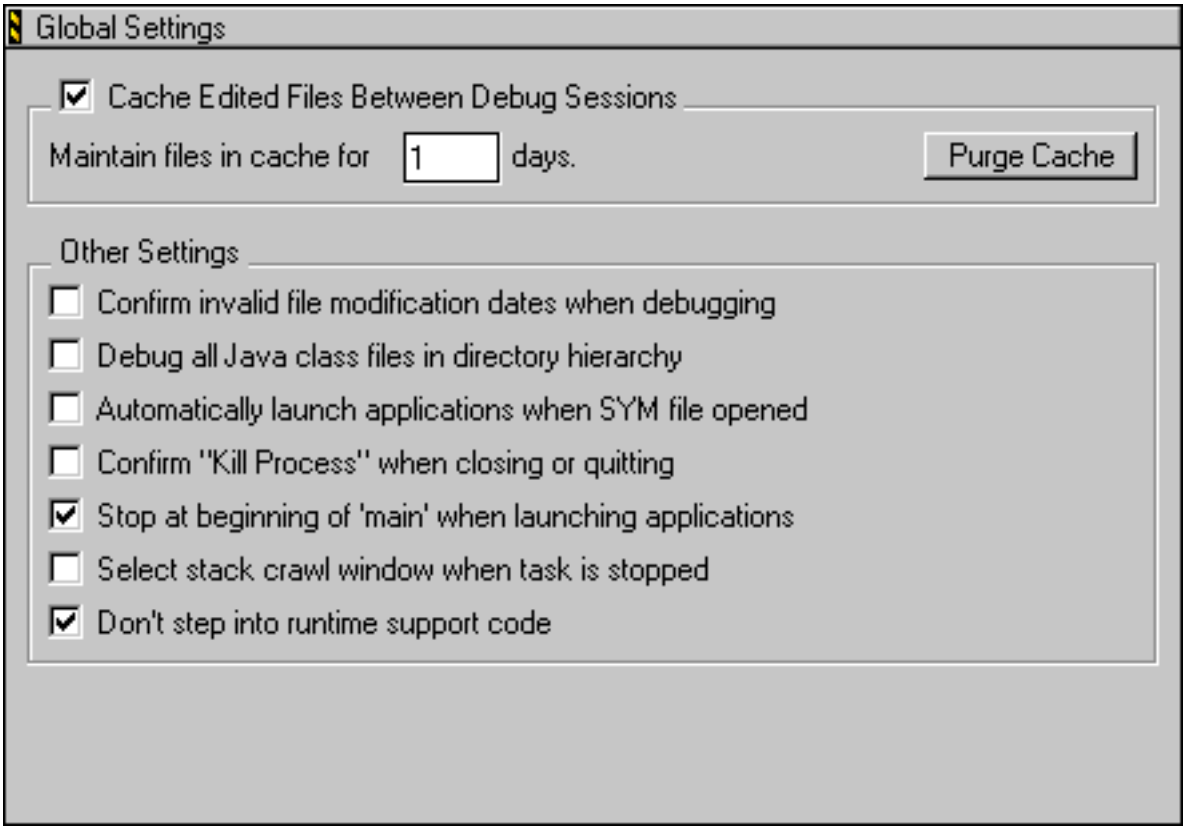
Open windows on debugging monitor during debugging

When you enable this option, any window that opens during a debugging session will appear on the selected debugging monitor. The IDE will not save a window's position on the debugging monitor if that window is closed during the debugging session. This prevents window positions from gravitating to the debugging monitor.

Global Settings Panel

This section tells you how to configure the Debugger's behavior to make your debugging chores easier. The Global Settings preference panel is shown in Figure 8.22.

Figure 8.22 Global Settings preference panel


Global Settings
Cache Edited Files Between Debug Sessions
Maintain files in cache for
1
days.
Purge Cache
Other Settings
Confirm invalid file modification dates when debugging
Debug all Java class files in directory hierarchy
Automatically launch applications when SYM file opened
Confirm "Kill Process" when closing or quitting
Stop at beginning of 'main' when launching applications
Select stack crawl window when task is stopped
Don't step into runtime support code

For more information on the commands contained in the Settings preference panel, see *Debugger User Guide*.

Cache Edited Files Between Debug Sessions

Enable this option to maintain a cache of edited files between debugging sessions. When enabled, you can specify the number of days to store the cache in the **Maintain files in cache for** text box. You can also reset the file caches using the **Purge Cache** control. When disabled, no file caches are maintained.

Confirm invalid file modification dates when debugging

For more information on this command, see the *Debugger User Guide*.

Debug all Java class files in directory hierarchy

This option applies when you want to debug a Java program but don't have a project file open.

When enabled during Java debugging, this option causes the debugger to search for additional symbolic files to open in the same folder as the class file opened and all its subfolders. When disabled, only the class file opened will be debugged.

Automatically launch applications when SYM file opened

Automatically launches a target program when its symbolics file is opened, setting an implicit breakpoint at the program's main entry point.

Confirm "Kill Process" when closing or quitting

Prompts for confirmation before aborting a process when a target program is killed.

Stop at beginning of 'main' when launching applications

When you begin debugging an application, the debugger stops at the first line of `main()`.

Select stack crawl window when task is stopped

Automatically brings the stack crawl window to the front when a task is stopped.

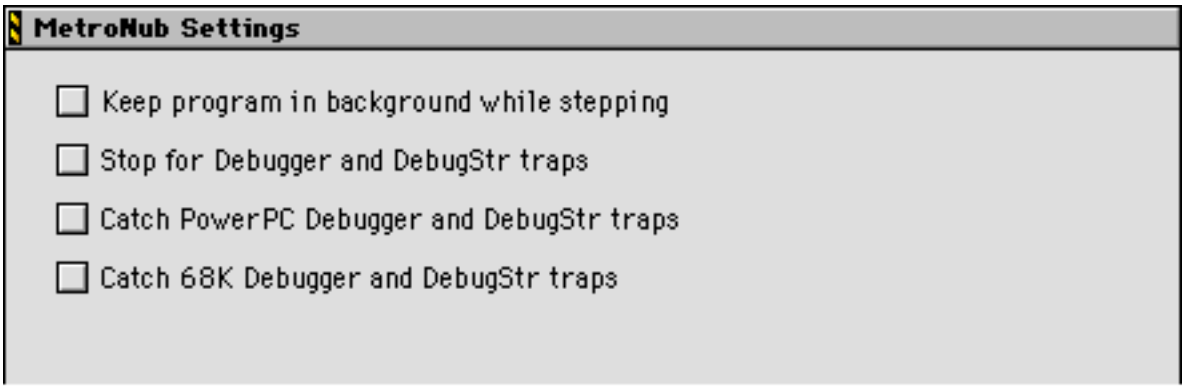
Don't step into runtime support code

Executes constructor code for C++ static objects normally, without displaying it in the program window.

MetroNub Settings Panel (Mac OS)

The MetroNub preferences is shown in Figure 8.23.

Figure 8.23 MetroNub preference panel (Mac OS)


MetroNub Settings
Keep program in background while stepping
Stop for Debugger and DebugStr traps
Catch PowerPC Debugger and DebugStr traps
Catch 68K Debugger and DebugStr traps

Keep Program in Background while Stepping

Use this option to keep the program being debugged behind all debugger windows. This makes it easier to observe variable changes and step through the source code without constantly switching between views. When disabled, the program operates in the foreground until a breakpoint or watchpoint is encountered and then the debugger will move in front of the program.

Stop for Debugger and DebugStr traps

Use this option to halt program execution whenever it encounters a Debugger or DebugStr trap. When disabled, both of these traps are ignored.

Catch PowerPC Debugger and DebugStr Traps

Use this option to halt program execution whenever it encounters any PowerPC Debugger or DebugStr traps. When disabled, both of these traps are ignored.

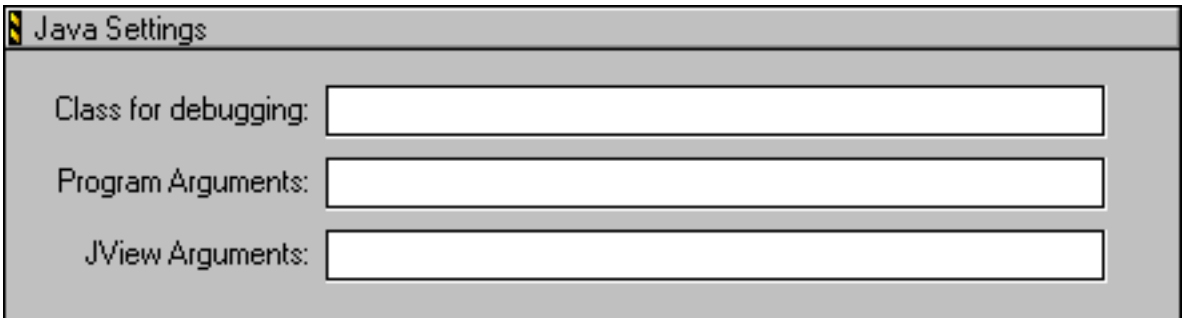
Catch 68K Debugger and DebugStr Traps

Use this option to halt program execution whenever it encounters any 68k Debugger or DebugStr traps. When disabled, both of these traps are ignored.

Java Settings Panel (Windows)

The Java Settings preference panel, shown in Figure 8.24, allows you to configure general Java debugger settings.

Figure 8.24 Java Settings preference panel



Class for Debugging

Use this text box to enter the class that you wish to debug.

Program Arguments

Use this text box to enter the arguments for the class you will debug. These parameters will be passed to the debugger.

JView Arguments

Use this text box to enter the arguments for the JView helper application.

x86 Settings Panel (Mac OS)

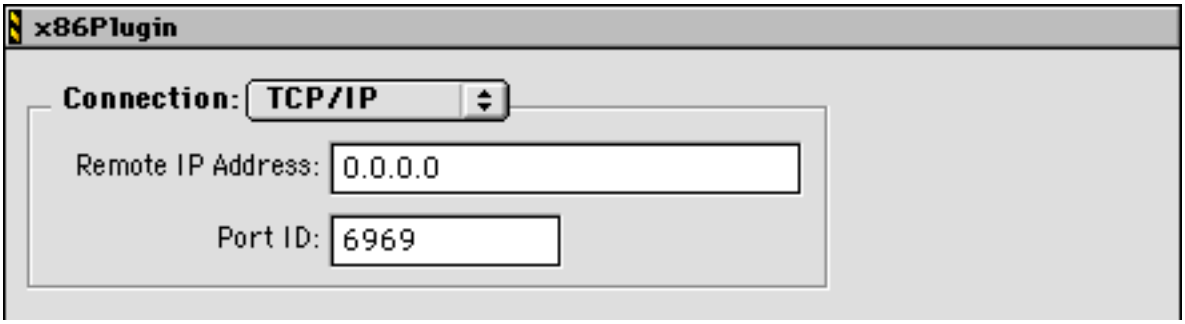
The x86 Settings preference panel, shown in Figure 8.25, allows you to set up your Mac OS computer to connect to a remote Windows computer for debugging sessions.

For more information about remote debugging, refer to the *Targeting Mac OS* manual.

Connection

Use this pop-up menu to set the type of connection between your computer and the remote computer. Your can choose from local port connections and TCP/IP connections. The options for TCP/IP connections are shown in Figure 8.25.

Figure 8.25 x86 Settings preference panel (Mac OS)



Remote IP Address

Use this option to specify the internet IP address of the remote computer to connect to for debugging. The IP address is used when you have a network connection to the remote computer.

Port ID

Use this option to specify the port ID number of the remote computer to connect to for debugging. The port ID number is used when you have a serial connection to the target computer.

NOTE: You should leave the port ID number at its default setting. Changing the default number may cause CodeWarrior to fail to connect to the remote computer.

Customizing Toolbars

A toolbar contains a series of elements, represented by icons, that act as buttons. Each element typically represents a corresponding menu command. When you click the element, that command is executed. There are other kinds of actions besides menu commands that can appear on a toolbar as well. Figure 8.26 shows the toolbar from the CodeWarrior Project window.

Figure 8.26 The Project window toolbar

TIP: (Mac OS) To display a balloon help window that describes a toolbar icon, place the cursor over the toolbar element and press the Control key.

This section discusses toolbars in detail. The topics are:

- Kinds of Toolbars
- Toolbar Elements
- Showing and Hiding a Toolbar
- Modifying a Toolbar
- Restoring a Toolbar to Default Settings
- Anchoring the Floating Toolbar (Mac OS)

Kinds of Toolbars

There are four toolbars in CodeWarrior:

- the floating toolbar (sometimes referred to as the global toolbar)
- the Project window toolbar, which appears in any project window
- the Editor window toolbar, which appears not only in the editor window, but also in the editing pane of other windows
- the Class Browser window toolbar, which appears in the single- and multi-class browser views

If you modify a toolbar, the changes apply to every instance of that toolbar subsequently created. For example, there is one Editor window toolbar that appears in all editor windows. If you modify the toolbar in one window, the change affects all editor windows. For more information, see "Modifying a Toolbar" on page 285.

Each of the toolbars has a factory-default configuration of elements that you can restore at any time. For more information, see "Restoring a Toolbar to Default Settings" on page 287.

There are really two groups of toolbars: the floating toolbar that can be used at all times, and toolbars that appear in particular windows.

This distinction is important, because you show, hide, clear, and reset the toolbars using menu commands in the Toolbar Submenu, which appears in the Window menu in the CodeWarrior IDE. These commands distinguish between the floating toolbar and window toolbars.

When you choose a menu command related to a window toolbar, the toolbar in the active window is affected. For more information, see "Toolbar Submenu" on page 408.

Toolbar Elements

A toolbar may contain any of four types of different elements. The four types of elements are:

- *Commands*—buttons that execute IDE menu commands when clicked

- *Controls*—the IDE popup menu buttons (Document Settings, Function, Header, Marker, Version Control), plus the Change Current Target button
- *Miscellaneous* —other elements (such as the File Dirty and File Path indicators)
- *Scripts* (Mac OS)—buttons that execute one of the scripts available through the Scripts menu in the IDE.

NOTE: (Windows) Scripting functionality has not yet been added to CodeWarrior Pro for Windows, so you cannot add script elements to a toolbar.

You access individual elements through the Toolbar Elements window, shown in Figure 8.27. To display this window, choose the **Toolbar Elements Window** item from the Toolbar Submenu of the Window Menu**.**

Each type of element has its own tab in the Toolbar Elements window. To see the list of available elements for any particular type, click the type name. For example, to see the list of available elements for toolbar commands, click the **Commands** tab in the Toolbar Elements window.

You use the Toolbar Elements window when adding new items to a toolbar. For more information on adding elements to a toolbar, see "Adding a Toolbar Element" on page 285.

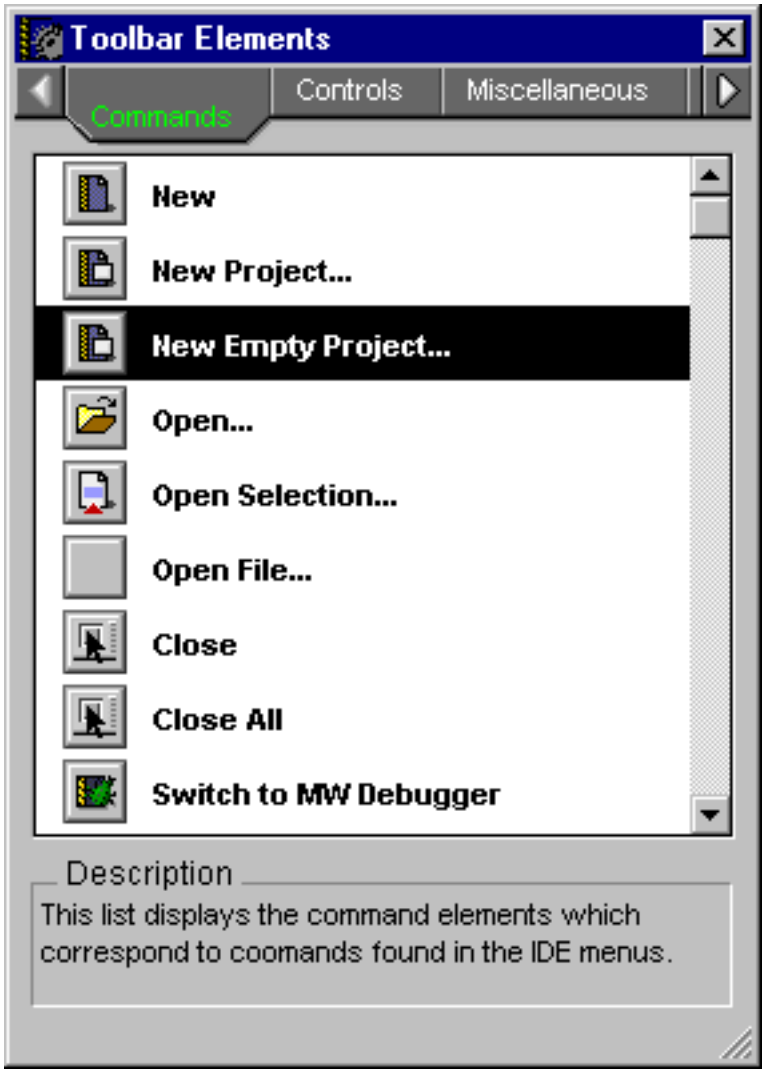
Showing and Hiding a Toolbar

Use Toolbar Submenu commands to show or hide a toolbar. Hiding a toolbar does not change the toolbar's configuration of elements.

Floating Toolbar

To display the floating toolbar, choose **Show Floating Toolbar**. To hide the floating toolbar, choose **Hide Floating Toolbar**.

Figure 8.27 The Toolbar Elements window



NOTE: Hiding the floating toolbar (on platforms that support it) does not change its anchored state. For example, if the floating toolbar is unanchored when hidden, it will remain unanchored when displayed again. For more information, see "Anchoring the Floating Toolbar (Mac OS)" on page 287.

Window Toolbar

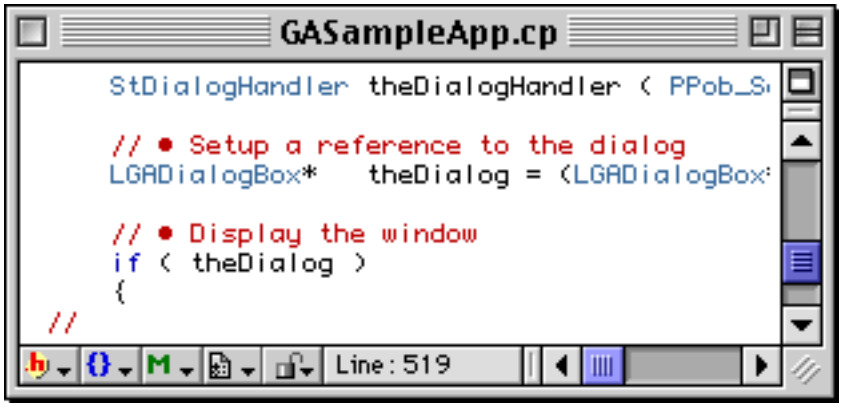
To hide a window's toolbar, first make sure that window is the active window. Then choose **Hide Window Toolbar** from the Toolbar submenu. The other components of the window shift to fill the space previously occupied by the window's toolbar. All subsequently opened windows of that type will have hidden toolbars.

To display a window's hidden toolbar, first make sure that window is the active window. Then choose **Show Window Toolbar**. The other components of the window shift to allow room for the window's toolbar. All subsequently opened windows of that type will display toolbars.

When you hide the Editor window toolbar, the default tools appear along the bottom of the editor window, as shown in Figure 8.28. When you show the Editor window toolbar, the tools reappear along the top of the editor window, as shown in Figure 5.1 on page 130.

You can also show or hide the Editor window toolbar with the Toolbar disclosure button. For more information, refer to "Seeing Window Controls" on page 138.

Figure 8.28 The Editor window with hidden toolbar



Modifying a Toolbar

You can modify a toolbar by:

- Adding a Toolbar Element
- Rearranging Toolbar Elements
- Removing a Toolbar Element
- Removing All Toolbar Elements

In certain circumstances there are limitations as to which elements you can add or remove from a toolbar. These are fully explained in "Adding a Toolbar Element" on page 285 and "Removing a Toolbar Element" on page 286.

If you modify a toolbar, the changes apply to every instance of that toolbar subsequently created. For example, if you customize the Project window toolbar, those changes will affect every Project window you open, not just the toolbar in the active Project window. Windows that are already open are not affected.

Adding a Toolbar Element

You add an element to a toolbar by dragging and dropping it from the Toolbar Elements window onto a toolbar.

Open the Toolbar Elements window, and locate the element you want to add to a toolbar. Make sure the destination toolbar is visible as well.

Drag the element from the Toolbar Elements window to the toolbar.

If the toolbar accepts the element, then framing corners will appear in the toolbar. These framing corners show you where the new element will appear when you release the mouse button. If for some reason you cannot add this element to this particular toolbar, no framing corners will appear.

There are several reasons why a toolbar may not accept an element:

- the toolbar is full
- the element already exists on the toolbar

- commands can be added to a window's toolbar only for menu commands that are available when the current window is the active window
- the five pop-ups on the Controls view of the Toolbar Elements window, as well as the File Dirty and File Path indicators on the Miscellaneous view, can only be added to the Editor window toolbar
- the Change Current Target element from the Controls view can only be added to the Project window toolbar

Rearranging Toolbar Elements

You can reposition toolbar elements to better suit your working requirements.

To position an element in a toolbar, first place the mouse pointer over the element. Next, press and hold Ctrl and the right mouse button (Windows) or press and hold Control, Command, and the mouse button (Mac OS). Then, drag the element to a new position. As you do, framing corners will appear in any location that can accept the drop. When you release the mouse button, the element will be repositioned inside the framing corners.

Removing a Toolbar Element

You can remove toolbar elements to better suit your working requirements.

To remove an element from a toolbar, Ctrl-right click (Windows) or Control-Command-click (Mac OS) on the element. The element is then removed from the toolbar.

Removing All Toolbar Elements

You can clear all elements from a toolbar using commands in the Toolbar Submenu of the Window menu. Clearing a toolbar is handy if you want to build your own toolbar from scratch.

To clear the floating toolbar, choose **Clear Floating Toolbar**.

To clear a window toolbar, make that window the active window. Then choose Clear Window Toolbar

In some cases, certain elements may not be removed by a Clear Window Toolbar command because they are critical to the window's basic purpose.

NOTE: If the floating toolbar is currently hidden, you cannot clear it. You'll have to display the floating toolbar before clearing it. See "Showing and Hiding a Toolbar" on page 282.

Restoring a Toolbar to Default Settings

You can reset a toolbar to its original factory settings (the program defaults) using commands in the Toolbar Submenu of the Window menu.

To reset the floating toolbar, choose **Reset Floating Toolbar**.

To reset a window toolbar, make that window the active window. Then choose Reset Window Toolbar.

NOTE: If the floating toolbar is currently hidden, you cannot reset it. You'll have to display the floating toolbar before resetting it. See "Showing and Hiding a Toolbar" on page 282.

Anchoring the Floating Toolbar (Mac OS)

The floating toolbar can be anchored to the top left corner of the screen, just below the menu bar. In its anchored state, the floating toolbar is joined to the IDE menu bar, loses the ability to be closed with a mouse-click, and cannot be moved. Unanchored, the floating toolbar can be positioned anywhere on your screen.

To anchor the floating toolbar when it is currently free-floating, choose **Anchor Floating Toolbar** from the Toolbar Submenu of the Window menu.

To release the floating toolbar when it is currently anchored, choose **Unanchor Floating Toolbar** from the Toolbar Submenu of the Window menu.

9

Configuring Target Options

This chapter discusses the Target Settings window. Target settings specify how the CodeWarrior IDE should process a build target in a project.

Configuring Target Options Overview

The Target Settings window handles settings that affect a particular build target in a project. To set options specific to the CodeWarrior build target within your current project, choose the **Target Settings** command from the Edit Menu. The actual name of this command will include the name of the build target.

In each case, the options are organized into a series of panels devoted to a particular topic. For example, one panel contains settings that specify the folders in which the IDE should search for the files listed in a build target's file view.

After you have set all the settings for a particular build target, you can create Project stationery so that your choices will be used when you create a new project. To learn more about this topic, refer to the discussion "Creating Your Own Project Stationery" on page 57.

The topics in this chapter include:

- Target Settings Guided Tour
- Choosing Target Settings

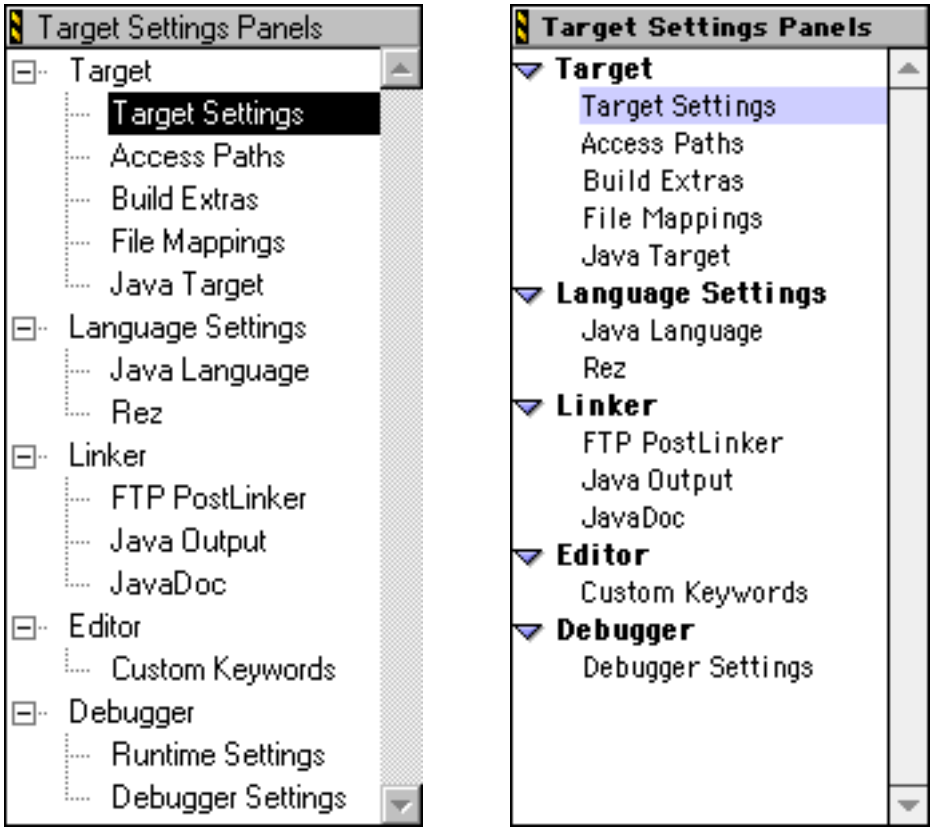
Target Settings Guided Tour

To open the Settings window, use the **Target Settings** command from the Edit Menu. When a project file is open, the actual name of the Target Settings command will include the name of the current build target. For example, if your current build target is **ANSI Console Win32**, then the Target Settings command will be named **ANSI Console Win32 Settings**.

The topics in this section are:

- Panels
- Dialog Box Buttons

Figure 9.1 Selecting a settings panel



Panels

The Target Settings window has a hierarchical list of available panels on the left side of the window. The panel selected in this list appears on the right side of the window. The actual panels available to you may vary, depending upon the CodeWarrior product you are using and the current build target.

To see a panel, select it in the list. You can use the arrow keys or click the name of the panel. Figure 9.1 on page 290 shows a selected panel in the Settings window.

Each panel consists of a series of related options that you set. The options apply to the currently selected build target in the active project. See the following section, "Dialog Box Buttons", for information on applying or ignoring changes to settings panels.

After you have set all the settings for a particular target, you can create Project stationery so that your choices will be used when you create a new project. To learn more about this topic, refer to the discussion "Creating Your Own Project Stationery" on page 57.

Dialog Box Buttons

There are several dialog box buttons in the Target Settings window that control how a panel's settings are used and applied.

The topics in this section are:

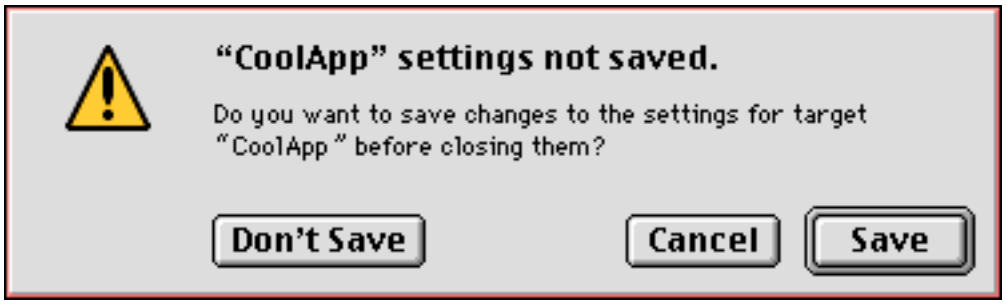
- Discarding Changes
- Factory Settings button
- Revert Panel button
- Save Button

Discarding Changes

If you make changes in the Target Settings window and attempt to close it, a dialog box similar to that shown in Figure 9.2 may appear. To save your changes and close the dialog box, click **Save**. To discard your changes and close the dialog box, click the **Don't Save**

button. To continue using the Target Settings window without saving changes, click **Cancel**.

Figure 9.2 Settings Confirmation Dialog Box


"CoolApp" settings not saved.
Do you want to save changes to the settings for target "CoolApp" before closing them?
Don't Save
Cancel
Save

Factory Settings button

The **Factory Settings** button causes the panel to revert to the settings that the CodeWarrior IDE uses as defaults. Settings in other panels are not affected by this button. Only the settings for the current panel are reset.

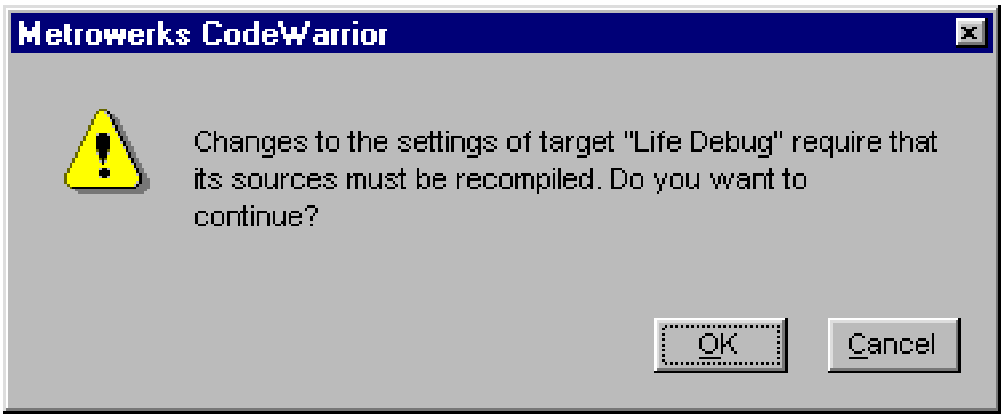
Revert Panel button

The **Revert Panel** button allows you to reset the state of the current panel you're viewing to its previously-saved settings. This is useful if you start making changes to a panel and then decide not to use them.

Save Button

The **Save** button commits any changes you made in any of the panels. If you changed an option that will require that the project be recompiled, you will see a dialog box similar to that shown in Figure 9.3. Click **OK** or **Cancel** depending on whether you want to keep your changes or not.

Figure 9.3 Save Changes Dialog Box


Metrowerks CodeWarrior
Changes to the settings of target "Life Debug" require that its sources must be recompiled. Do you want to continue?
OK
Cancel

After closing the Target Settings window, the CodeWarrior IDE will handle target builds according to the settings that you saved.

Choosing Target Settings

This section discusses setting options for a particular target. You can change many different settings to configure the CodeWarrior IDE to build your target the way you want.

To learn how to open the Settings window and select a particular settings panel, see "Target Settings Guided Tour" on page 290.

There are many settings panels available for customizing CodeWarrior Professional. The particular panels that appear in the Settings window depend upon the particular operating system or chip family you have set as your target, and the programming language that you use.

For example, if you are working with an x86 project, you will not see any panels for Motorola 68K configurations. If the Pascal language is not available for a particular target, panels related to Pascal will not appear.

In addition, the IDE takes great care to ensure that only the files affected by a preference option are marked for recompilation. For example, changing a preference in the resource compiler only marks

resource compiler sources as dirty. This reduces the amount of re-compilation required during the next Make operation and increases linking speed.

This manual does not discuss panels that are specific to a particular operating system, microprocessor family, or language. Table 9.1 lists the various manuals where you can learn the details of OS-, processor-, or language-specific settings panels. This list is not exhaustive, and does not include every target or panel supported in various CodeWarrior products. This list represents the panels provided with CodeWarrior Professional.

While some panels are specific to a particular language, operating system, or microprocessor family, other panels control options that are pertinent to any target. These panels are discussed in detail in this section. They are:

- Target Configurations
- Editor Configurations

Table 9.1 Where to learn about specific settings panels

For these panels	See this manual
x86 Target Windows RC x86 Processor FTP Postlinker x86 Linker x86 Exceptions	*Targeting Win32*
Java Project Java VM Java Linker	*Targeting Java*
68K Target Rez 68K Processor 68K Disassembler 68K Linker CFM68K Linker	*Targeting Mac OS*

For these panels	See this manual
PPC Target PPCAsm Rez PPC Processor PPC Disassembler FTP Postlinker PPC Linker PPC PEF	*Targeting Mac OS*
C/C++ Compiler C/C++ Warnings	*C Compiler Guide*
Pascal Compiler Pascal Warnings	*Pascal Compiler Guide*
Runtime Settings Settings	*Debugger User Guide*

Target Configurations

The following panels apply to the CodeWarrior targets:

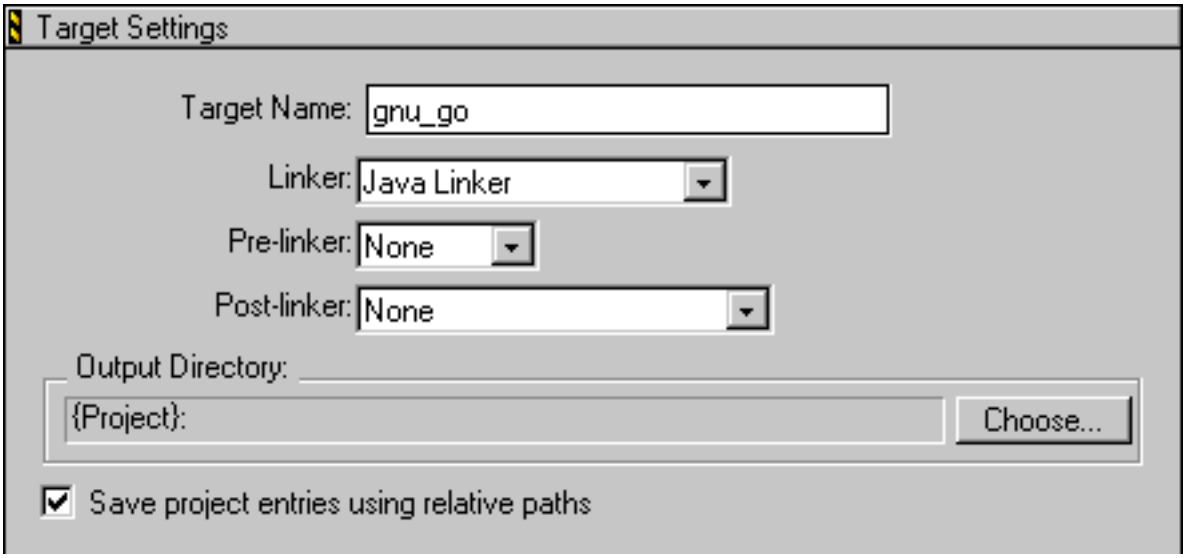
- Target Settings
- Access Paths
- Build Extras
- File Mappings

Target Settings

The Target Settings panel is the single most critical panel in CodeWarrior. This is the panel where you pick your target operating system and/or microprocessor.

To learn how to open the Settings window and select the Target Settings panel, see "Target Settings Guided Tour" on page 290.

Figure 9.4 Target Settings panel



The Target Settings panel, shown in Figure 9.4, allows you to set the name of your target, as well as the linker and post linker plug-ins for the target. When you select a linker, you are specifying the target operating system and/or chip. The other panels available in the Settings window will change to reflect your choice.

Because the linker choice affects the visibility of other related panels, you must set your target first before you can specify other target-specific options like compiler and linker settings.

It is possible to change targets completely in this panel. When you change the target, you must change the libraries contained in the project file for that target. Choosing a new value for a target does not change these files for you. For this reason, you should be careful when changing the target to remove inappropriate libraries and add those required for the new target.

If you create a new project from stationery appropriate for the new target, the necessary library and support files will be included automatically. You can then add source files from the old project. For more information on creating a project based on stationery, see "Creating a New Project" on page 50.

Target Name

Use the Target Name text field to set or change the name of a target. When you use the Targets view in the Project window, you will be able to see the name that you have set.

This is not the name of your final output file, just the name you assign to the target for your personal use. The name of the final output file is typically set in the linker settings panel for the linker you choose for your target.

Linker

Choose a linker from the items listed in the Linker pop-up menu.

To learn more about the choices available to you for the linker and post linker, refer to the *Targeting* manual for your selected target. The actual choices available to you will depend upon the plug-in linkers you have for your CodeWarrior product.

For example, CodeWarrior Professional provides linkers for Mac OS on 68K and PowerPC, Win32 on x86, and the Java Virtual Machine. CodeWarrior for PlayStation OS provides a linker for the PlayStation OS running on MIPS, as well as a post-linker.

NOTE: See "File Mappings" on page 307 for more information about the file mappings associated with the linker that you choose. These file mappings determine whether the IDE will recognize files in the project.

Post Linker

Some targets have post linkers that perform additional work (such as a data format conversion) on the final executable. Refer to information in the *Targeting* manual for your target.

Output Directory

This is the directory where your final linked output file is placed. The default location is the directory that contains your project file. Click the **Choose** button to specify another directory.

Save Project Entries Using Relative Paths

When this setting is enabled, the IDE remembers the location of a project entry as a relative path from one of the access paths. This extra location information lets the IDE distinguish different source files with the same name. The IDE will remember this location even if it needs to re-search for files in the access paths.

If this setting is disabled, then the IDE remembers project entries by name and re-searching for files could cause the IDE to find the project entry in a different access path.

See "Re-search for Files" on page 393 and "Reset Project Entry Paths" on page 393 for more information.

Access Paths

If you need to define additional access paths for the CodeWarrior IDE to search while compiling and linking your project, use the Access Paths settings panel, shown in Figure 9.5 (Windows) and Figure 9.6 (Mac OS).

To learn how to open the Settings window and select the Access Paths panel, see "Target Settings Guided Tour" on page 290.

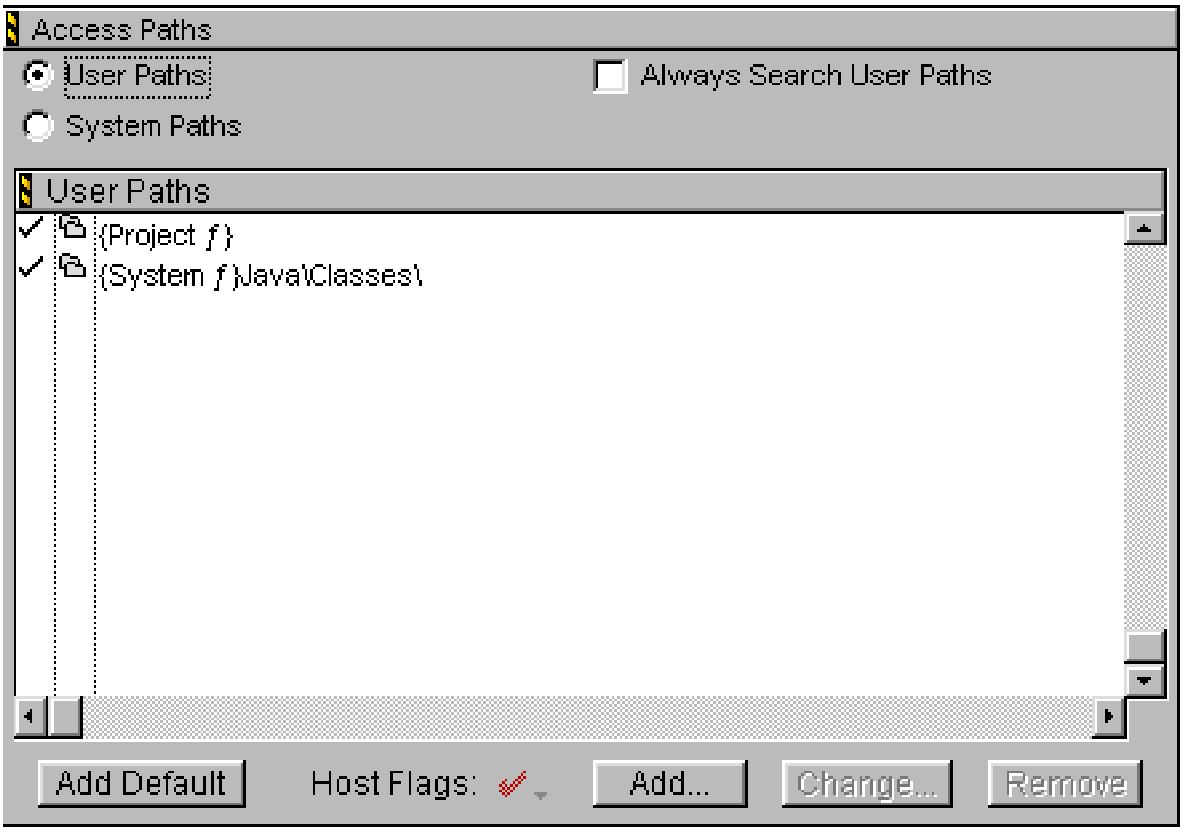
If a folder icon appears beside the name of a folder in either the User Paths pane or the System Paths pane, the CodeWarrior IDE performs a recursive search on the path. That is, the CodeWarrior IDE searches that folder and all the folders within it.

By clicking the folder icon to the left of any path in the User Paths pane or the System Paths pane, you can disable recursive searching of all subdirectories below that path. If the folder is visible, recursive search is turned on. If the folder is not visible, all subdirectories of that path will not be searched by the compiler.

TIP: If you turn off recursive searching of paths, and add each specific path of every directory that contains your files to either the System or User Path panes, you will speed compilation of your project.

By clicking the check mark icon to the left of any path in the User Paths pane or the System Paths pane, you can disable searching that path from the current CodeWarrior host computer. If the access path is checked, the current host computer will search that path. If the access path is not checked, the current host computer will ignore that path.

Figure 9.5 Access Paths settings panel (Windows)



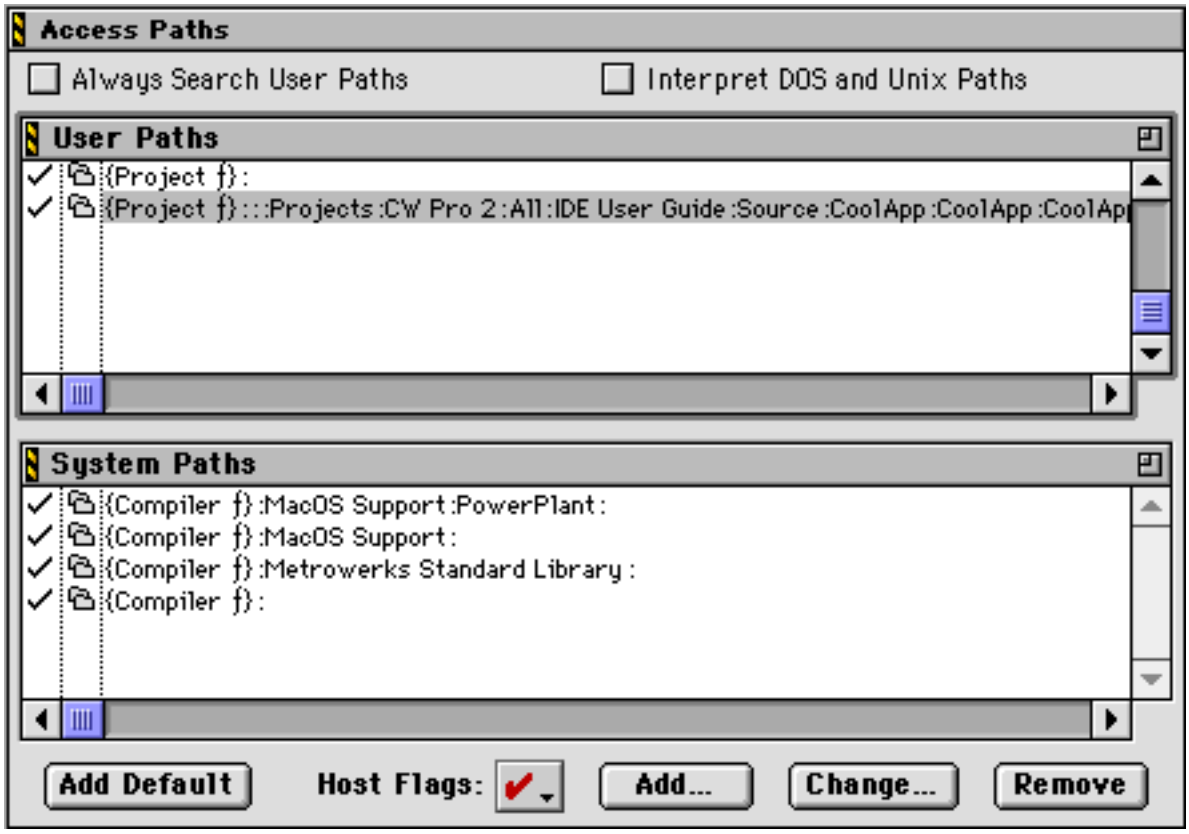
If your project's files or libraries are not in either of the default access paths, the CodeWarrior IDE will not find them when compiling, linking, or running your project. You must add their access path to tell the IDE where to look.

For more information about access paths, see "Interfaces pop-up menu" on page 106.

Windows `Resource.frk` files are also automatically excluded from the search list.

TIP: You can prevent any folder and all its subfolders in an access path from being searched by renaming it with enclosing parentheses. For example, changing `GameImages` to `(GameImages)` will exclude it from all subsequent searches. To add it to the search list, it must be explicitly added as an access path.

Figure 9.6 Access Paths settings panel (Mac OS)



Always Search User Paths

To search for system header or interface files in the same way as user header files, turn on this option.

User Paths (Windows)

Displays the User Paths pane in the Access Paths preference panel.

System Paths (Windows)

Displays the System Paths pane in the Access Paths preference panel.

Interpret DOS and Unix Paths (Mac OS)

Determines how the IDE should treat file names for interface files. When this checkbox is deselected, the IDE treats "\" and "/" characters as simply part of the file name in an interface file. If this checkbox is selected, the IDE treats these characters as sub-folder separator characters.

For example, with this checkbox deselected, the IDE treats this directive,

```
#include "sys/socks.h"
```

to mean that a file named "`sys/socks.h`" should be searched for. If the checkbox is selected, then the IDE looks for a sub-folder named "`sys`" that has a file named "`socks.h`".

User Paths pane

In Pascal, these access paths are searched first. In the C language, an `#include "..."` statement searches these access paths. By default, it contains `{Project}`, which is the folder that contains the open project.

System Paths pane

In Pascal, these access paths are searched after those in the User Include Path Pane. In C, an `#include < ... >` statement searches these access paths. By default, it contains `{Compiler}`, which is the folder that contains the CodeWarrior IDE.

Add Default

The CodeWarrior IDE lets you add the default path for the User Paths pane or System Paths pane after you have deleted the default

path. To add the default path to the access path pane you are working with, click the **Add Default** button. the CodeWarrior IDE adds the default path back into the relevant path pane.

Host Flags

Specifies the host platform that can use an access path. To enable a host platform to use a particular access path, select the access path in the User Paths pane or System Paths pane, then choose the host platform from the Host Flags pop-up menu.

To specify that the access path may be used no matter which host platform the IDE runs on, choose **All** from the pop-up menu.

For example, selecting a path and choosing **Mac OS** in the Host Flags pop-up menu specifies that the IDE must search the access path when the IDE is running on a Mac OS computer and ignore the access path when running on any other platform.

Add

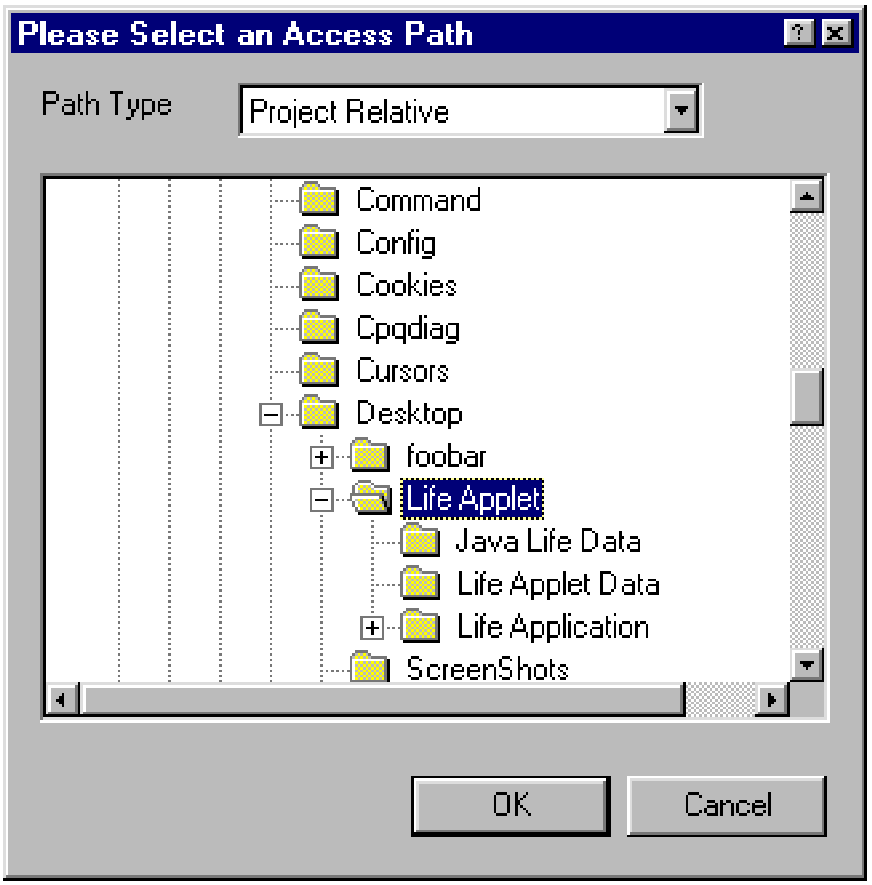
To add a new access path, first select the User Paths pane or the System Paths pane, then click the **Add** button. The dialog box shown in Figure 9.7 (Windows) or Figure 9.8 (Mac OS) appears. You can also use drag-and-drop to add paths to the Access Paths settings panel.

You can specify how CodeWarrior stores an access path by choosing one of these path type options: Absolute Path, Project Relative, Compiler Relative, and System Relative.

NOTE: Relative paths allow projects to contain two or more files with identical names. However, for large projects, you may notice slower performance when adding relative paths to a project.

Absolute Path means that the IDE stores the access path from the root level of the startup hard drive to the folder whose access path you want to add, including all folders in between. You need to update absolute access paths if you move the project to another system, rename the hard drive, or rename any of the folders along the access path.

Figure 9.7 Access Path Selection Dialog Box (Windows)



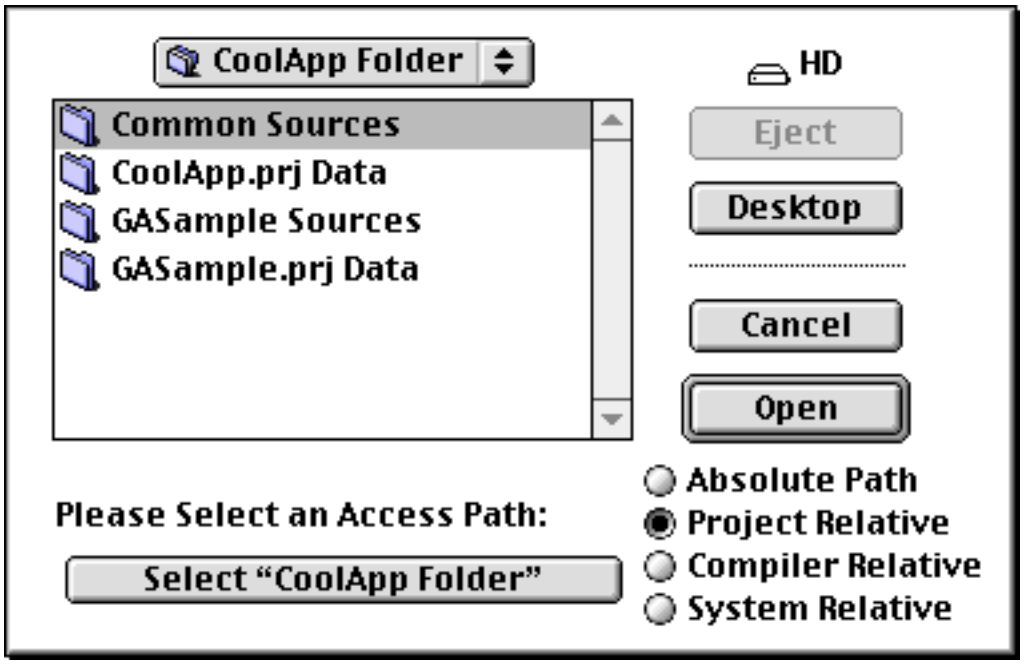
Project Relative means that the IDE stores the access path from the folder that contains the project to the folder whose access path you want to add. You do not need to update project relative access paths if you move a project, as long as the hierarchy of the relative path is the same. You cannot create a relative path to a folder on a different hard drive than where your project file resides.

Compiler Relative means that the IDE stores the access path from the folder that contains the CodeWarrior IDE to the folder whose access path you want to add. You do no need to update compiler relative access paths if you move a project, as long as the hierarchy of the relative path is the same. You cannot create a relative path to a folder on a different hard drive than where your CodeWarrior IDE resides.

System Relative means that the IDE stores the access path from the operating system's base folder to the folder whose access path you want to add. You do not need to update system relative access paths if you move a project, as long as the hierarchy of the relative path is the same. You cannot create a relative path to a folder on a different hard drive than where your active operating system's base folder resides.

Select an access path type from the four available types, and then select the folder whose access path you want to add. Click **OK** to have the CodeWarrior IDE add the access path to the list, or click **Cancel** to leave the list unchanged.

Figure 9.8 Access Path Selection Dialog Box (Mac OS)



NOTE: (Mac OS) Beginning with Mac OS 8.5, the access path selection dialog box will no longer appear as shown in Figure 9.8. Instead, you will see an improved dialog box. Choose the desired access path type from the four options under **Path Type**. When you select a folder in the list and click **Open**, you will see the con-

tents of that folder. When you locate the folder whose access path you wish to add, click **Choose**. The access path to that folder will appear in the access paths pane.

Change

To change an access path, first select a path in the System Paths Pane or the User Paths Pane. Then click the **Change** button. The dialog box shown in Figure 9.7 on page 303 (Windows) or Figure 9.8 (Mac OS) appears. Use this dialog box to navigate to the location of the access path you want to change to.

To learn more information about the options in the dialog box, refer to "Add" on page 302.

Remove

To remove an access path, first select the path to be removed from the System Paths Pane or the User Paths Pane. Then, click the **Remove** button, and the path is removed.

Windows You can also remove access paths by dragging them to the Recycle Bin on the Desktop.

Mac OS You can also remove access paths by dragging them to the Trash on the Desktop.

Build Extras

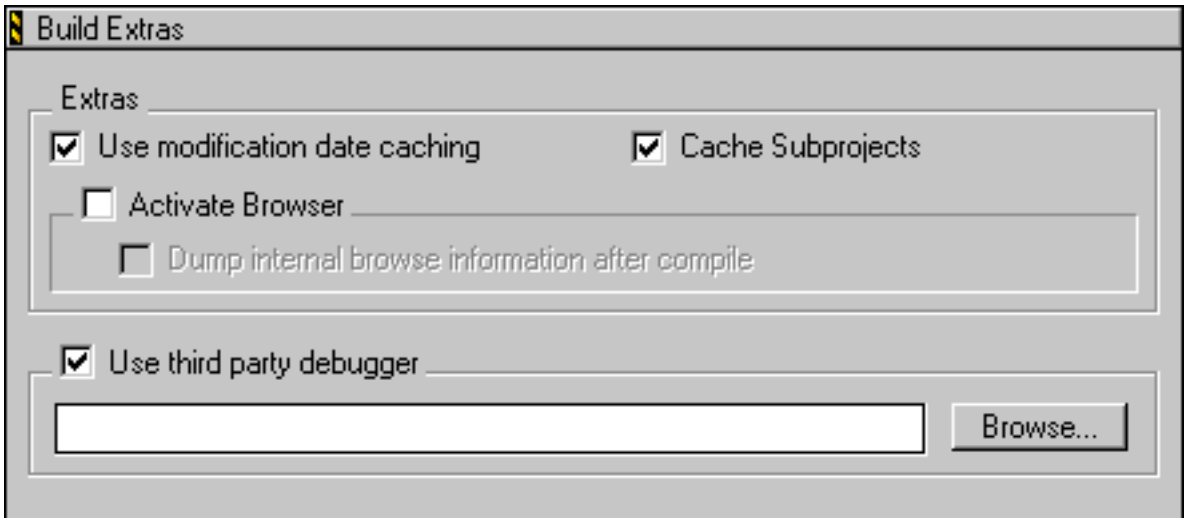
The Build Extras panel contains various options that affect the way a project builds. These options are shown in Figure 9.9 (Windows) and Figure 9.10 on page 307 (Mac OS).

To learn how to open the Settings window and select the Build Extras panel, see "Target Settings Guided Tour" on page 290.

Use Modification Date Caching

Before making a project, the CodeWarrior IDE checks the modification dates of the files to see if you've changed them outside the CodeWarrior IDE. If you edit files with the CodeWarrior Editor only, turn on the **Use Modification Date Caching** option to shorten compilation time.

Figure 9.9 Build Extras settings panel (Windows)



Activate Browser

Selecting this option causes the information needed by the CodeWarrior code browser to be generated the next time your project is built. Without this information, you cannot open Browser windows for your project.

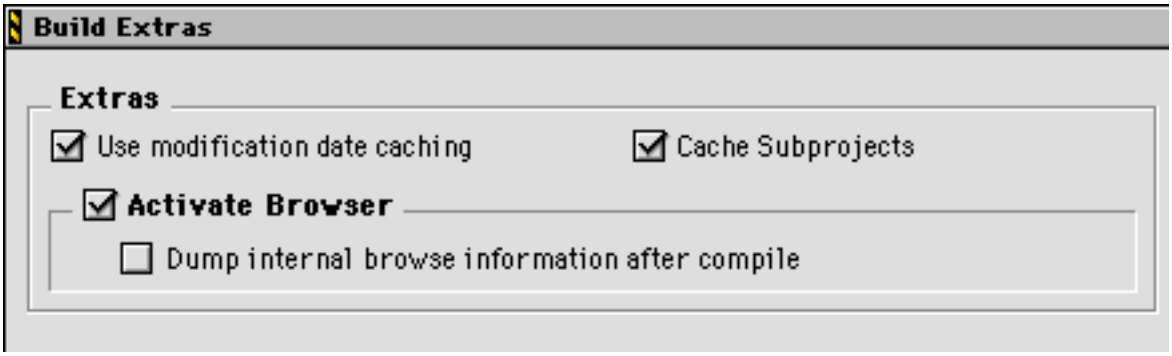
See "Making a Project" on page 323 for more information about re-building your project. For more information on browser settings and options, see "Browser Overview" on page 201.

Cache Subprojects

Selecting this option improves multiproject updating and linking. Selecting this option also allows the IDE's class browser to include browser information from a target's subprojects as well as the target's own browser information.

Deselecting this option reduces the amount of memory required by the CodeWarrior IDE.

Figure 9.10 Build Extras settings panel (Mac OS)



Dump Internal Browse Information After Compile

Select this option to view the raw browser information that a plug-in compiler or linker provides for the IDE. This option's only practical use is for developing plug-ins for the IDE.

NOTE: When this option is on, compile only single files, or small files. The information that the IDE displays when this option is on can be huge when compiling an entire project.

Use Third Party Debugger (Windows)

Enable this option to use a third-party debugger in place of the CodeWarrior internal Debugger. Enter the path to the debugger application in the field provided, or click the Browser button to locate the debugger application of your choice using the standard open file dialog.

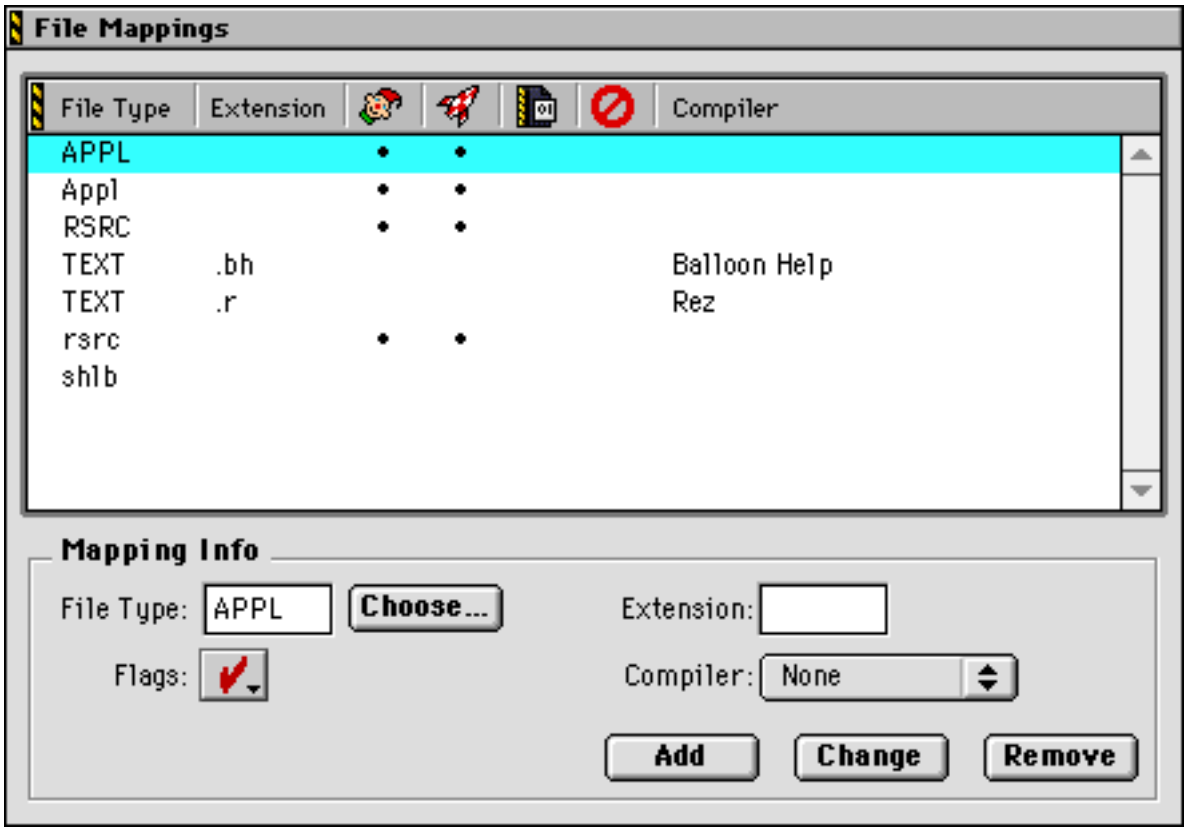
File Mappings

The File Mappings settings panel, shown in Figure 9.11, is used to associate a file name extension such as `.c` or `.p` with a plug-in compiler. This tells the CodeWarrior IDE which compiler to use when a file with a certain name is encountered.

NOTE: These file mappings determine whether the IDE will recognize files in the project. If you have trouble adding files to your project, or if the IDE refuses a folder or file that is dragged and dropped on the project window, check the File Mappings settings panel. The file mappings are associated with the currently selected linker, and those mappings will change as you select different linkers. For more information, see "Linker" on page 297

To learn how to open the Settings window and select the File Mappings Panel, see "Target Settings Guided Tour" on page 290.

Figure 9.11 File Mappings settings panel



File Mappings List

The File Mappings List contains a **File Type**, associated **Extension**, and compiler choice for each file name extension in the list. This list tells the CodeWarrior IDE which compiler to invoke when a given file name is encountered.

To add a new extension to this list, choose an existing entry in the list, edit the **Extension** and **Precompiled** fields, and click the **Add** button. You can choose values for the **Flags** before clicking the **Change** button.

To add documentation files to your project, choose the `.doc` Extension, delete the File Type, delete the compiler selection, and click **Add**.

Extension

This flag allows you to enter a file name extension such as `.c` or `.h` for a File Type you are working with in the File Mappings List.

Table 9.2 shows the default file name extensions used by the CodeWarrior IDE.

Table 9.2 Default File Name Extensions

Type	Extension	Description
Minimum CodeWarrior Installation	`.iSYM`	CodeWarrior Intel Symbols
	`.mch`	CodeWarrior Precompiled Header
	`.mcp`	CodeWarrior Project File
	`.SYM`	CodeWarrior Mac OS 68K Debug Symbols
	`.xSYM`	CodeWarrior Mac OS PPC Debug Symbols

Type	Extension	Description
Default CodeWarrior Installation	`.dbg`	CodeWarrior Debug Preferences
	`.exp`	Exported Symbol File
	`.iMAP`	CodeWarrior Link Map
	`.MAP`	CodeWarrior Link Map
	`.xMAP`	CodeWarrior Link Map
Library	`.lib`	Library File
	`.o`	Object File
	`.obj`	Object File
	`.pch`	Precompiled Header Source File
	`.pch++`	Precompiled Header Source File
Default C and C++	`.c`	C Source File
	`.cp`	C++ Source File
	`.cpp`	C++ Source File
	`.h`	C and C++ Header File
Default Java	`.class`	Java Class File
	`.jar`	Java Archive File
	`.jav`	Java Source File
	`.java`	Java Source File
Default Pascal	`.p`	Pascal Source File
	`.pas`	Pascal Source File
Assembly	`.a`	Assembly Source File
	`.asm`	Assembly Source File
	`.dump`	CodeWarrior Disassembled File

Type	Extension	Description
C and C++	`.c++`	C++ Source File
	`.cc`	C++ Source File
	`.hh`	C++ Header File
	`.hpp`	C++ Header File
	`.i`	C Inline Source File
	`.icc`	C++ Inline Source File
	`.m`	Object C Source File
	`.mm`	Object C++ Source File
Java	`.JMAP`	Java Import Mapping Dump
	`.jpob`	Java Constructor File
	`.mf`	Java Manifest File
Pascal	`.ppu`	Pascal Precompiled Unit

Compiler

This field allows you to choose a compiler for a File Type you are working with in the File Mappings List.

Flags

This pop-up menu allows you to enable and disable four options:

Resource File This flag indicates that the CodeWarrior IDE will include the resources from these files in your finished product.

Launchable This flag indicates that the CodeWarrior IDE will open the source code file with the application that created it when you double-click it in a project window.

Precompiled This flag indicates that the CodeWarrior IDE will compile these files before other files. This is useful if these files create documents that other source files or compilers use. For example, this option lets you create a compiler that translates a file into a C

source code file and then compile the C file. YACC (Yet Another Compiler Compiler) files are treated as precompiled files since YACC generates C source code to be compiled by a C compiler.

Ignored by Make This flag indicates that the CodeWarrior IDE will ignore these files when compiling or linking the project. This is useful if the files contain comments or documentation that you want to include with your project.

Editor Configurations

The following panels apply to the CodeWarrior Editor:

- Custom Keywords

Custom Keywords

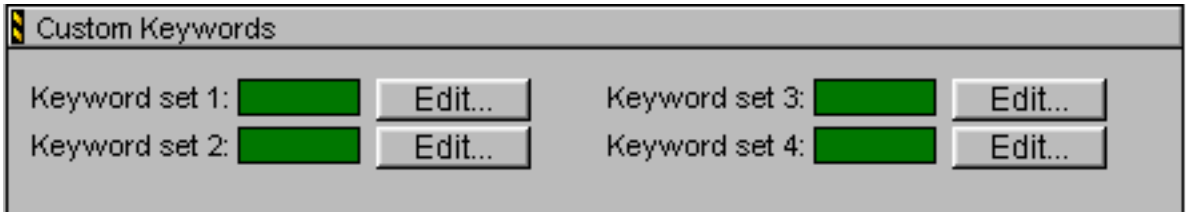
The Custom Keywords settings panel, shown in Figure 9.12, allows you to define your own keyword sets that have certain colors associated with them when they appear in your Editor files. These keywords are project-specific, not global to the CodeWarrior IDE.

To learn how to open the Settings window and select the Custom Keywords panel, see "Target Settings Guided Tour" on page 290.

For information on setting global keyword sets, see "Syntax Coloring" on page 266. That topic also discusses the details of setting colors, specifying keywords, and importing and exporting keywords.

To change the color for a keyword set, click the color sample. To change the contents of a keyword set, click the **Edit** button and make the appropriate entries in the dialog box that appears.

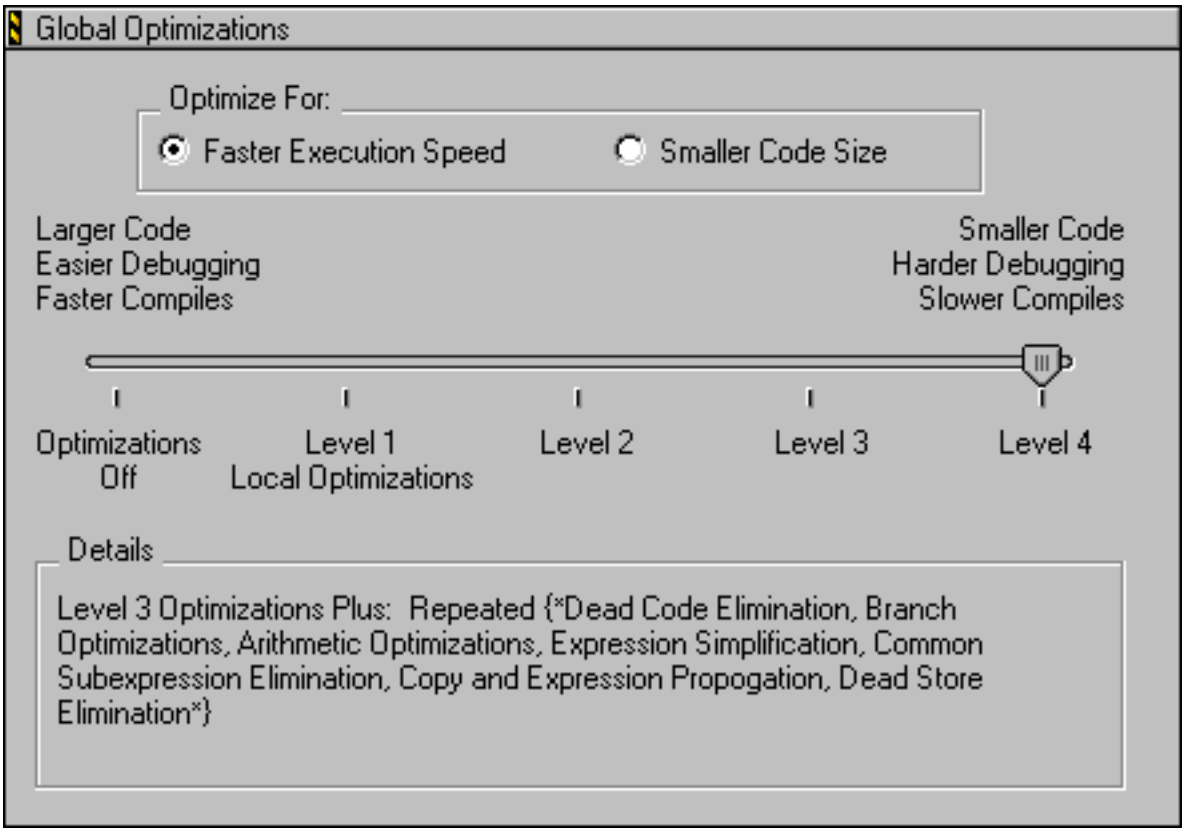
Figure 9.12 Custom Keywords settings panel



Global Optimizations

The Global Optimizations panel, shown in Figure 9.13, is available when targeting many platforms. Use this panel to instruct the compiler to rearrange its object code to produce smaller and faster-executing object code. Some optimizations remove redundant operations in a program, other optimizations analyze how an item is used in a program and attempt to reduce its affect on a program's performance.

Figure 9.13 Global Optimizations settings panel



To learn how to open the Settings window and select the Global Optimizations panel, see "Target Settings Guided Tour" on page 290.

All optimizations rearrange object code without affecting the object code's logical sequence of execution. In other words, an unopti-

mized program and its optimized counterpart produce the same results.

NOTE: Use compiler optimizations only after you've debugged your software. Using a debugger on a program that has been optimized may affect the debugger's source code view.

To find specific information on how these settings apply to a target, see "Targeting Documentation" on page 24.

Optimize For

Use these options to configure how the CodeWarrior IDE optimizes your code.

Faster Execution Speed This option improves the execution speed of object code. Object code is faster, but may be larger.

Smaller Code Size This option reduces the size of object code that the compiler produces. Object code is smaller, but may be slower.

Optimization Level Slider

Use the slider to determine the optimizations that are applied to your code. You can choose to turn off code optimizations, or you can choose to apply one of four levels of optimizations. The higher the level that you select, the more optimizations are applied to your code. The Details text field, below the slider, lists the optimizations that are applied. The following list describes those optimizations:

Global Register Allocation This option also appears as **Global Register Allocation Only for Temporary Values** when you choose Level 1 optimizations.

Dead Code Elimination Removes statements that, logically, can never be executed or are never referred to by other statements. With this option on, object code is smaller

Common Subexpression Elimination Replaces similar redundant expressions with a single expression. For example, if two consecutive statements both use the expression a * b * c + 10, the compiler generates object code that computes the expression only once

and applies the resulting value to both statements. With this option on, object code is smaller and faster.

Copy Propagation Replaces multiple occurrences of one variable with a single occurrence. With this option on, object code is smaller and faster. This option also appears as **Copy and Expression Propagation** for some build targets.

Peephole Optimization Applies local optimizations to small sections of your code. With this option, the optimized sections of code are faster.

Dead Store Elimination Removes assignments to a variable if the variable is not used before being reassigned again. With this option on, object code is smaller and faster.

Loop-Invariant Code Motion Moves computations that don't change on the inside of a loop to the outside of the loop to improve the loop's speed. With this option on, object code is faster.

Strength Reduction Replaces multiplication instructions that are inside loops with addition instructions to speed up the loop. With this option on, object code is larger but executes faster.

Loop Unrolling This option is optimized for speed only.

Lifetime Based Register Allocation Uses the same processor register for different variables in the same routine if the variables aren't used in the same statement. With this option on, object code executes faster.

10

Compiling and Linking

This chapter discusses how to compile, link, and run a CodeWarrior project.

Compiling and Linking Overview

The information in this chapter assumes you have already created a project, added the necessary files, grouped these files, and set the project's options. To learn more about these tasks, refer to other chapters in this book, including "Working with Projects Overview" on page 41, "Working with Files Overview" on page 101, "Source Code Editor Overview" on page 129, and "Configuring IDE Overview" on page 235.

You should also be familiar with features such as moving files in the Project window, the Project window's columns, and Project window pop-up menus. To learn more about these features, refer to "Working with Projects Overview" on page 41.

This chapter discusses how to compile and link a project to produce your code, and how to correct common compiler and linker errors using the Message window. It does not describe in detail the various types of programs the CodeWarrior IDE can create. For that information, please see the *CodeWarrior Targeting* manual appropriate for your platform. A table describing which guide to refer to is shown in "Targeting Guides for Various CodeWarrior Targets" on page 25.

The CodeWarrior IDE can only compile and link files belonging to an open project. That is, you should have a project open before trying to compile any files.

The topics in this chapter are:

- Choosing a Compiler
- Compiling and Linking a Project
- Using Precompiled or Preprocessed Headers
- Preprocessing Source Code
- Disassembling Source Code
- Guided Tour of the Message Window
- Using the Message Window
- Special Library Options (Mac OS)

Choosing a Compiler

When you create source code files, you are using a certain programming language such as C, C++, Pascal, or another language. These languages have naming conventions for the files. For example, in the C language, a source code file ends with a `.c` suffix and a header file ends with a `.h` suffix.

This section describes how to associate a file suffix with a compiler for a given language in the CodeWarrior IDE.

Understanding Plug-in Compilers

The CodeWarrior IDE is designed to allow compilation of many different programming languages. In order to make the product modular to accept many different languages, plug-in compilers are used.

Plug-ins are basically small loadable code modules that allow the IDE to have many different compilers at its disposal. For example, there are plug-in compilers for C, C++, Pascal, Java, and assembly language.

Plug-in compilers usually have default target settings to help the CodeWarrior IDE decide which project files a plug-in handles. During regular compile and link operations, the IDE assigns files to the proper plug-in automatically.

Setting a File Extension

To associate a plug-in compiler with a given file, you set the File Mappings options. For a description of how to configure these options to set a compiler for your source code files, refer to "File Mappings" on page 307.

Compiling and Linking a Project

The CodeWarrior IDE provides many different ways to build a project. When you build a project, you compile and link it.

All compiling and linking commands are available from the Project Menu. Depending on your project type, a few of these commands may be disabled or renamed. For example, you cannot Run a shared library, but you can Make it. Also, a compiling or linking command may be dimmed because CodeWarrior is busy executing another command or the project is being debugged.

The CodeWarrior IDE can only compile and link files belonging to an open project. That is, you should have a project open before trying to compile any files.

If you have multiple projects open at the same time, you may want to learn about how to select a default project before compiling and linking. To learn about this topic, refer to "Choosing a Default Project" on page 67.

The topics in this section are:

- Compiling Files
- Setting Link Order
- Updating a Project
- Making a Project
- Enabling Debugging
- Running a Project
- Debugging a Project
- Generating a Link Map

- Synchronizing Modification Dates
- Removing Objects
- Advanced Compile Options

Compiling Files

You can tell CodeWarrior to compile a single file or certain files in your project. Of course, CodeWarrior can also compile all of the files in your project.

You may want to switch between multiple targets in a project when compiling files. To learn how to do this, refer to "Setting the Current Build Target" on page 91.

The **Compile** command on the Project Menu is dimmed when:

- There is no open project.
- The active Editor window does not have a source code file name extension.
- The active window's source code file is not included in your project.

Windows Only

- The binary file, such as the application you created from the project, is running under the integrated debugger.
- An application created from the project is running.

As CodeWarrior compiles source code files and libraries in the open project, it highlights them in the Project window. The Build Progress window (Figure 10.1) displays a line count and the name of the file being compiled.

A status line displaying the total number of files to be compiled and the number of files being compiled is also provided at the bottom of the Project window.

Figure 10.1 Build Progress window

Compiling One File

To compile a single file in your project, select that file in your Project window and choose **Compile** from the Project Menu.

Alternatively, you can open the file in the CodeWarrior Editor, make the window the active window, and choose Compile from the Project Menu.

Compiling Selected Files

You may select several files in your project for compilation by selecting the files in the Project window and then choosing **Compile** from the Project Menu. To learn how to select several files in your project, refer to "Selecting Files and Groups" on page 70.

Recompiling Files

CodeWarrior doesn't always recognize file changes and may not automatically recompile the file. You can force CodeWarrior to recompile a changed file. To do this, you must touch the file first. To learn how to touch a file, refer to "Touching and Untouching Files" on page 83.

After touching the file or files that you want to recompile, choose **Bring Up To Date** or **Make** from the Project Menu.

Setting Link Order

You can specify the order in which files are compiled in the Link Order view of the Project window. By re-arranging the order of the files, it's possible to prevent linkage errors caused by file dependencies. That is, if file `Aleph` depends upon file `Zeta` already being compiled, you can position file `Zeta` to compile before file `Aleph`, preventing an error message.

To Set the Link Order:

1. **Click the Link Order tab in the Project window.**

   The Link Order view appears in the Project window.

2. **Drag files into the correct link order.**

   Use drag-and-drop to reposition the files into the desired build order.

The next time you **Bring Up To Date**, **Make**, **Run**, or **Debug** the project, the new build order will be used when compiling the project files.

See "Link Order View" on page 49 for additional information.

NOTE: (Mac OS) If you are building a 68K project, the Link Order tab is replaced by the Segments tab.

Updating a Project

When you have many newly added, modified, or touched files in your project, you can use the **Bring Up To Date** command on the Project Menu to compile all the files.

When using this command, the linker will not be invoked, so your project will not have its output binary produced. This command only runs the compiler.

You may want to switch between multiple build targets when updating a project. To learn how to do this, refer to "Setting the Current Build Target" on page 91.

Making a Project

When you are ready to produce your binary file, such as an application, library, or shared library, use the **Make** command on the Project Menu. This command builds the selected project type by updating the newly added, modified, and "touched" files, then linking the project.

The results of a successful build depend on the selected project type. For example, if the project type is an application, the Make command builds an application and saves it in the same folder as your project. Table 10.1 lists some example project types and what is built when the Make command is executed. To find a full list of the types of software products you can produce, refer to the targeting guide for your target. Refer to Table 1.1 on page 16 to find out which guide to read for your build target.

Table 10.1 Example Products for Certain Targets

Project Type	Target	Make Creates
Application	Win32	Win32 application
Library	Win32	Win32 Library (`.lib`)
Dynamic link library	Win32	Win32 Dynamic Link Library (`.dll`)
Application	68K/PowerPC	Mac OS Application
Library	68K/PowerPC	Mac OS A4 or A5 Library
Code resource	68K/PowerPC	Code Resource
MPW tool	68K/PowerPC	MPW Tool
Shared library	68K/PowerPC	PowerPC Shared Library

Once all the modified files and "touched" files have been compiled successfully, CodeWarrior links all the files in the project to produce your output binary. If the project has already been compiled using **Bring Up To Date** or another command, then the **Make** command only links the compiled source code files together.

You may want to switch between multiple targets in a project when making a project. Refer to "Setting the Current Build Target" on page 91 for more information.

Enabling Debugging

When the **Enable Debugger** option is chosen in the Project Menu, choosing the **Debug** command lets the integrated debugger launch and debug your project. When you choose **Disable Debugger** from the Project Menu, choosing the **Run** command runs your project normally.

To configure your project so that files in the project have debugging information generated for them, refer to "Controlling Debugging in a Project" on page 97.

To learn more about running your project, refer to the following section, entitled "Running a Project."

Running a Project

When you choose the **Run** command from the Project Menu, CodeWarrior compiles and links the project (if necessary), creates a stand-alone application, and then launches that application.

If the current project is not executable, the Run command is not available. Non-executable projects include libraries, shared libraries, dynamic linked libraries, code resources, or tools.

When compiling and linking is successful, CodeWarrior saves the new application on your hard disk. It is named according to options you set. If you would like to change these options, refer to "Choosing Target Settings" on page 293.

If the current project is designed to operate on another platform, you must connect your host computer to the target computer or device before choosing Run. For additional information, see the appropriate targeting manual for the target platform.

Debugging a Project

To debug your project, there are basically two steps you need to do. Of course, you must already have your project compiled and linked with debugging information generated. To learn how to enable debugging for your project, refer to "Enabling Debugging" on page 324.

The second step is for you to start the integrated debugger with your compiled application as the debug target. You can do this by choosing the **Debug** command from the Project Menu. If the Debug command is not in the menu, you do not have debugging enabled for your project. Refer to "Enabling Debugging" on page 324 to learn how to remedy this.

Once you have chosen the Debug command, CodeWarrior compiles and links your project, creates a debugging information file, and then opens that debug information file with the integrated debugger.

If the Debug command is dimmed, make sure the proper options for debugging are configured, as detailed in "Enabling Debugging" on page 324. Also, make sure that integrated debugger application is on your hard disk. If Debug is still dimmed, you are probably attempting to run a project whose project type cannot be run (such as a shared library or library), or the application is already running.

NOTE: The Debug command does not open any application that you may need to debug your project. If you're debugging an Adobe Photoshop Plug-in or any other project that requires an application, you must launch the application on your own before you choose Debug.

If the current project is designed to operate on another platform, you must connect your host computer to the target computer or device before choosing Run. For additional information, see the appropriate targeting manual for the target platform.

Once you are in the integrated debugger, you need to refer to the *CodeWarrior Debugger Manual* for information on how to use it.

TIP: If you want to switch to the Debugger while you are in a CodeWarrior Editor window, use the Switch to MW Debugger (Mac OS) command on the File Menu.

Generating a Link Map

The CodeWarrior C/C++ compilers let you create a link map file that contains function and class section information on the generated object code.

CodeWarrior's Pascal compilers let you create a make map file that contains a list of dependencies and the compilation order.

The settings to control this option are in the target's Linker panel. After configuring your project, you will need to Make your project. If the compile and link is successful, a link map file named after your project with the `.MAP` extension is saved in your project folder.

To learn more about a target's Linker panel, see the discussion in the appropriate targeting manual.

Synchronizing Modification Dates

If you want to update the modification dates stored in the project file for all files in your project, choose the **Synchronize Modification Dates** command from the Project Menu.

To learn more information about this topic, see "Synchronizing modification dates" on page 84.

Removing Objects

When you compile your project, the CodeWarrior IDE adds the object code from each source file to the project. This binary object code increases the size of the project file. There are a few different commands available if you want your project file to consume less memory on the hard disk, or you want to remove all object code and start compilation over again.

Removing Object Code

In some cases, you may wish to remove all the object code from the project and restart the compiling and linking process. To remove a project's object code, select the **Remove Object Code** command from the Project Menu. The dialog box shown in Figure 10.2 appears.

Figure 10.2 Remove Objects dialog box

Remove object code from project "CoolApp.prj".
Removing object code from a target will require it to be rebuilt. Do you want to remove the object code from only the current target or from all targets in project "CoolApp.prj"?
All Targets
Cancel
Current Target

Clicking **All Targets** removes all object code data for all build targets in the project, resetting the Code and Data size of each file in the project window to zero. **Current Target** removes the objects for the current build target only, and leaves the objects in place for all other build targets. **Cancel** aborts the operation so none of your object code is removed.

To learn how to change the current build target of your project, refer to "Setting the Current Build Target" on page 91.

Remove Objects & Compact command

This command removes all binaries from the project and compacts it to consume the minimum amount of space on your hard disk. The procedure is similar to that discussed in "Removing Object Code" on page 327, but in addition the project file is compressed.

Compacting the project removes all object code and debugging information stored in the project file and retains only the information about the files that belong to the project and project-specific option settings.

For more information about this command, refer to "Remove Object Code & Compact" on page 392.

Advanced Compile Options

This section describes two options that either speed up your project build times or alert you when a build is completed.

Alerting Yourself After a Build (Mac OS)

You may start a project compile/link cycle in CodeWarrior, then switch to another application running on your machine. To learn how to receive notification when the build is completed, refer to "Play Sound After 'Bring Up To Date' & 'Make' (Mac OS)" on page 242.

Speeding Up a Build by Avoiding Date Checks

To learn about how to optimize the speed of your builds in CodeWarrior, refer to "Use Modification Date Caching" on page 305.

Using Precompiled or Preprocessed Headers

Source code files in a project typically use many header files (“.h” or “.hpp” files). Often, the same header file is included in many different source code files, forcing the compiler to (inefficiently) read the same header files many times during the compilation process. Many programming languages support precompiled headers, including C and C++.

To shorten the time spent compiling and recompiling a header file, use the **Precompile** command on the Project Menu. A precompiled header file takes the compiler significantly less time to process than an ordinary, uncompiled header file.

For instance, a header file that contains the most frequently used headers in your project could be made into one precompiled header file. Instead of having to compile the same thousands of lines of header files for each source file in your project, the compiler only has to load one precompiled header file.

NOTE: You can only include one precompiled header in a source file. Including more than one precompiled header will result in an error.

TIP: The precompiled header format frequently changes in order to accommodate new features in CodeWarrior updates. Therefore, precompiled header formats are often incompatible between CodeWarrior updates. After installing a new CodeWarrior update, you usually need to precompile your precompiled headers to use the new format. However, you can also configure your precompiled headers to automatically update to the new format. See “Automatic updating” on page 331 for more information.

The topics in this section are:

- Creating Precompiled Headers
- Defining Symbols For C/C++

- Defining Symbols For Pascal

Creating Precompiled Headers

To create a precompiled header file, you must first open a project. The option settings from this project are used when precompiling. A file to be precompiled does not have to be a header file (`.h` or `.hpp`), but it must meet these requirements:

- The file must be a text file. You cannot precompile libraries or other files.
- The file must be saved with a recognized extension, such as `.pch` or `.pch++`, at the end of its file name.
- The file does not have to be in a project, although a project must be open to precompile.
- The file must not contain any statements that generate data or code. However, C++ source code can contain inline functions and constant variable declarations (`const`).
- Precompiled header files for different targets are not interchangeable. For example, to generate a precompiled header for use with Win32 compilers, you must use a Win32 compiler.
- A source file can include only one precompiled header file using the `#include` directive.

To create a precompiled header file, create a text file using **New** on the File Menu. In that text file, put your #include directives. For example, if you want to create a precompiled header file of the files `string.h` and `stdio.h`, just put the following in your text file:

```
#include <stdio.h>
#include <string.h>
```

Once the text file is saved, the IDE can use the text file to create a precompiled header. You can specify the name of the precompiled header at the time of precompilation. You can also specify the name by including the following line at the beginning of your text file, where *name* is the name you wish to call the precompiled header:

```
#pragma precompile_target "name"
```

Windows Append the extension `.mch` to *name*. This extension is required for precompiled headers.

Precompile command

To precompile a saved text file, choose **Precompile** from the Project Menu. This command precompiles the text file in the active window, creating a precompiled header file. If compiler errors are detected, a Message window appears.

To learn more about the Message window and correcting compiler errors, consult "Correcting Compiler Errors and Warnings" on page 343.

To learn more about automatic updating of precompiled headers, see the next section, "Automatic updating."

Automatic updating

If the source code has been modified, the CodeWarrior IDE automatically updates a precompiled header during a **Make** or **Bring Up To Date** operation.

If the CodeWarrior IDE encounters a `.pch` or `.pch++` file in the project that was modified since it was last precompiled, the CodeWarrior IDE precompiles it again to ensure that the resulting precompiled header is up-to-date.

To create a precompiled header file that is automatically updated, open the project that will use the precompiled header. Then create a text file that will be used to create the precompiled header.

To read about the requirements for a precompiled header text file, refer to "Creating Precompiled Headers" on page 330.

In the first line of the text file, add the line:

```
#pragma precompile_target "name"
```

This `pragma` tells the compiler to create a precompiled header with the file name of *name*.

Windows Append the extension `.mch` to *name*. This extension is required for precompiled headers.

When you're finished with the text file, save it with a valid extension in the same folder as your project.

Windows Save the text file with a `.pch` extension if it will be used to create a precompiled header in either C or C++.

Mac OS Save the text file with a `.pch` extension if it will be used to create a precompiled header in C. Use the `.pch++` extension if the text file will be used to create a precompiled header in C++.

Then choose **Precompile** from the Project Menu.

Now add the text file to the open project with the **Add Window** command in the Project Menu.

Whenever the precompiled header file is modified, the CodeWarrior project manager will automatically update it by precompiling it.

To include the precompiled header in a project source code file, add this line as the first #include directive in the source code file:

```
#include "name"
```

Alternatively, you can specify a precompiled header as a prefix file, using the settings for your target. To learn about this, refer to the discussion of the C/C++ Language settings panel in the *C Compiler Guide*.

NOTE: Do not use the precompiled header text file (`.pch` or `.pch++`) in `#include` directives; use the name of the resulting precompiled header file instead (`.mch`). Although using the precompiled header text file is legal and will not affect the final binary, you won't be taking advantage of the precompiled header's speed.

Defining Symbols For C/C++

To automatically update and add predefined symbols and other preprocessor directives, you can create a precompiled header file, add it to your project, and make it the prefix file specified in the C/C++ Language settings panel.

1. **Create a new text file.**

   Open your project and create a new text file with the **New** command on the File Menu.

   This new text file will contain your preprocessor directives. You'll use this file as a precompiled header file that you will add to your project.

2. **Open the C/C++ Compiler settings panel.**

   Choose **Target Settings** from the Edit Menu. The actual name of the Target Settings command will include the name of the target. Select the **C/C++ Language** settings panel in the Target Settings window.

3. **Get the Prefix File name if it exists.**

   If there is a file name in the Prefix File box, Copy it, then close the Target Settings window.

   In the new text file window, paste the file name that you copied from the Prefix File box in an include directive. Make sure this is the *first* directive in the text file.

   For example, if the Prefix File is `MyHeaders`, then the first directive in the text file window is:

   ```
   #include <MyHeaders>
   ```

4. **Add the `#pragma` statement.**

   Add the `#pragma precompile_target` statement to the text file. This statement lets you name the precompiled file. For example, to create a file named `MyPrecomp`, use the following statement:

   ```
   #pragma precompile_target "MyPrecomp"
   ```

   Windows Append the extension `.mch` to the desired file name. This extension is required for precompiled headers. In this example, the file name must be: "`MyPrecomp.mch`"

5. **Type in all your own preprocessor directives.**

   Type in all your own `#define`, `#include`, and other preprocessor directives corresponding to the needs of your source code. The text file cannot contain any source code that generates data or executable code. However, C++ source code can contain inline functions and constant variable declarations (`const`).

6. Save **the text file.**

   Choose **Add Window** from the Project Menu to add the text file to your project. Save the text file with a valid extension in the same folder as your project.

   Windows Save the text file with a `.pch` extension.

   Mac OS Save the text file with a `.pch` extension if it will be used to create a precompiled header in C. Use the `.pch++` extension if the text file will be used to create a precompiled header in C++.

7. Precompile **the text file.**

   Choose **Precompile** from the Project Menu. The IDE will use the text file to create a precompiled header file.

8. **Open the C/C++ Compiler settings panel.**

   Choose **Target Settings** from the Edit Menu and select the **C/C++ Language** settings panel.

9. **Set the new Prefix File.**

   In the Prefix File field, enter your precompiled file's name, in this example "`MyPrecomp`". Click **Save** to save your changes.

   Windows In this example, you would enter "`MyPrecomp.mch`" in the Prefix File field.

   Whenever your project is built, the CodeWarrior project manager updates your precompiled header and automatically includes it in each source code file.

Defining Symbols For Pascal

Although the Pascal preprocessor is not as powerful as the C/C++ preprocessor, you can still create files that can automatically insert your own preprocessor symbols and compiler directives into your

project. For more information on the Pascal compiler directives, see the Pascal Language Manual on the CodeWarrior Reference CD.

1. **Create a new text file.**

   Open your project and create a new text file with the **New** command.

   This new text file will contain your compiler directives.

2. **Open the Pascal Compiler settings panel.**

   Choose **Target Settings** from the Edit Menu. The actual name of the Target Settings command will include the name of the target. Select the **Pascal Language** settings panel in the Target Settings window.

3. **Get the Prefix File name if it exists.**

   If there is a file name in the Prefix File box, Copy it, then close the Target Settings window.

   In the new text file window, paste the file name that you copied from the Prefix File box in an include directive, `{$I}`. Make sure this is the *first* directive in the file.

   For example, if the Prefix file is `OtherDefs.p`, then the first directive in the editor window is:

   ```
   {$I OtherDefs.p}
   ```

4. **Type in all your own {$SETC}, {$I}, and other preprocessor directives.**

   The text file cannot contain any source code that generates data or executable code.

5. Save **the text file.**

   Save it as an ordinary Pascal file in the same folder as your project. For example, save this file as "`MyPrecomp.pas`".

6. Precompile **the text file.**

   Choose **Precompile** from the Project Menu. The IDE will use the text file to create a precompiled header file.

7. **Open the Pascal Compiler settings panel.**

   Choose **Target Settings** from the Edit Menu, and select the **Pascal Language** settings panel.

8. **Set the new Prefix File.**

   In the Prefix File field, enter your file's name, in this example "`MyPrecomp`", then click **Save** to save your changes.

   Whenever your project is built, the CodeWarrior project manager automatically includes the precompiled header in each source code file.

Preprocessing Source Code

The preprocessor prepares source code for the compiler. It interprets directives beginning with the "`#`" and "`$`" symbols (such as `#define, $pragma` and `#ifdef`), removes extra spaces and blank lines, and removes comments (such as `/*…*/` and `//`). You might want to preprocess a file if you want to see what the code looks like just before compilation.

Open a file that you want to preprocess, or select a file in your currently-open Project window. To preprocess a file, select the **Preprocess** command on the Project Menu. The results of the Preprocess command are stored in a new file named after the source code file that was preprocessed and beginning with the "`#`" character.

To save the contents of the new window, choose one of the save commands in the File Menu.

Disassembling Source Code

If you wanted to see the code that would be generated for your file you could disassemble the file. Disassembling is useful if you want to know the machine-level code that is being executed when your source code is executed. In addition, the disassembled code can be a model for writing your own assembly routines. Library files may also be examined using this command.

The **Disassemble** command on the Project Menu disassembles the compiled source code file selected in the project window and displays its assembly-language code in a new window. The title of the

new window consists of the name of the source code file with the extension "`.dump`."

To save the contents of the "`.dump`" window, choose one of the save commands in the File Menu.

If the file being disassembled has not been compiled, the disassemble command will compile the file before disassembling it.

Guided Tour of the Message Window

The Message window, shown in Figure 10.3, is used to display messages about events that have occurred when compiling, linking, or searching files. There are a number of elements in the window that are useful for accomplishing certain tasks, such as navigating to error locations and scrolling to see all messages for a project.

There are some user interface items in the Message window that are not discussed here. To learn about the Marker Pop-Up Menu, Options Pop-Up Menu, VCS Pop-Up Menu, File Path Caption and Line Number Button, refer to "Guided Tour of the Editor Window" on page 130.

The topics in this section include:

- Error Button
- Warning Button
- Project Information Caption
- Extra Information Button
- Stepping Buttons
- Message List Pane
- Source Code Disclosure Triangle
- Source Code Pane
- Pane Resize Bar

Error Button

The Error Button in the Message window toggles the view of error messages on and off. This is useful if you have changed the view of the window to something else and want to get back to viewing the error messages.

To learn more about seeing error messages in the Message window, refer to "Seeing Errors and Warnings" on page 341.

Warning Button

The Warning Button in the Message window toggles the view of warning messages on and off

To learn more about seeing error messages in the Message window, refer to "Seeing Errors and Warnings" on page 341.

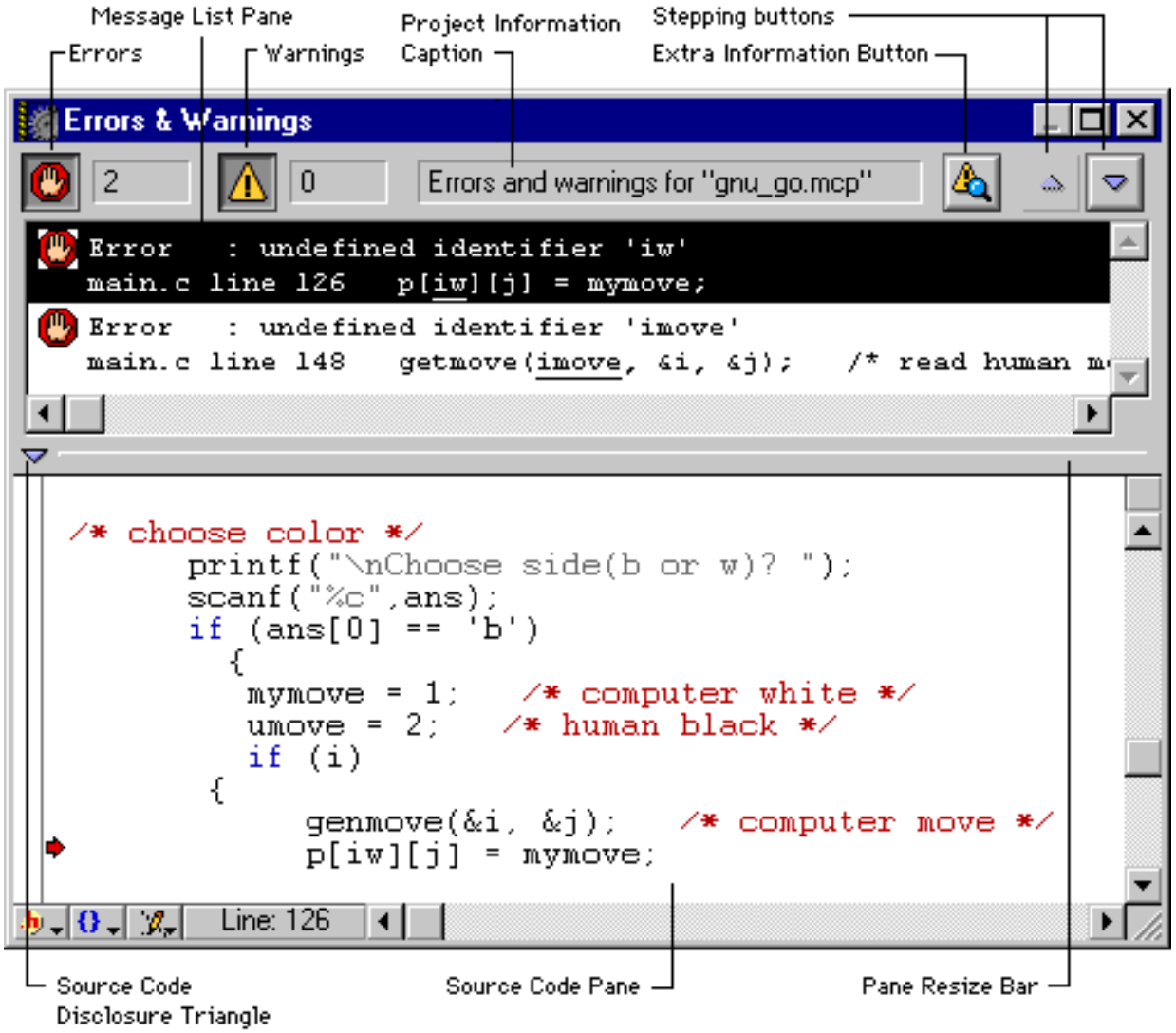
Project Information Caption

The Project Information Caption gives a short description of the view you are looking at in the Message window. Your project name will appear here.

Extra Information Button

The Extra Information Button expands a message to show information about the project, target, and file that caused the message.

Figure 10.3 The CodeWarrior Message Window



Stepping Buttons

The Stepping Buttons allow you to step up or down through the messages in the window.

To learn more about stepping through messages in the Message window, refer to "Stepping Through Messages" on page 342.

Message List Pane

The Message List Pane displays your messages.

To learn more about seeing messages in the Message window, refer to "Seeing Errors and Warnings" on page 341.

Source Code Disclosure Triangle

The Source Code Disclosure Triangle allows you to hide the Source Code Pane of the Message window.

Source Code Pane

The Source Code Pane of the Message window allows you to view the source code at the location referred to by a message. To learn more about the view in this window, refer to "Seeing Errors and Warnings" on page 341.

Pane Resize Bar

The Pane Resize Bar allows you to reallocate the amount of space in the Message window given to the Source Code Pane and Message List Pane. By clicking and dragging this bar up or down you will change the amount of space on your computer screen that is allocated to both panes.

Using the Message Window

While compiling your project, the CodeWarrior IDE may detect a syntax error or other type of compiler error in one of your project's source code files. If this happens, the Message window displays the total number of errors and warnings, and information about each one. See "Guided Tour of the Message Window" on page 337 for information on the interface items in this window.

In this section, you will learn how to interpret, navigate, and use the information that appears in the Message window. The topics in this section include:

- Seeing Errors and Warnings

- Stepping Through Messages
- Correcting Compiler Errors and Warnings
- Correcting Linker Errors
- Correcting Pascal Circular References
- Saving and Printing the Message Window
- Locating Errors in Modified Files

Seeing Errors and Warnings

The Message window displays several types of messages:

Select this …	To display this…
Errors	Either compiler or linker errors. Both types of errors prevent the compiler and linker from creating a final binary.
Warnings	Either compiler or linker warnings. Neither type prevents the CodeWarrior IDE from creating a binary. However, they indicate potential problems during run time. You can specify which conditions lead to warning messages or you can upgrade all warnings to errors.
Notes	All other types of messages issued in the Message window. For example, results of a batch find are notes messages.

To close the Message window, click its close box or select **Close** in the File Menu while the Message window is the active window. If you close the message window and want to see it again, choose the **Errors & Warnings Window** command from the Window Menu to reopen it.

To see only error messages in the Message List Pane, click on the Error Button and turn off the Warning Button.

To see only warnings in the Message List Pane, click the Warning Button and turn off the Error Button.

To see both errors and warnings in the Message List Pane, click both buttons. Notes do not appear in the Errors & Warnings window.

You'll also see other types of messages from time to time in a Message window, such as:

- During **Add Window** or **Add Files** when a file being added does not reside on an existing access path.
- During linking when a project contains conflicting resources.
- During a Find when the Batch checkbox is selected in the Find dialog box.

Mac OS Only Messages

- During a **Find Reference (Mac OS)** when more than one definition for the same function is found (multiple definitions).
- During a **Find Reference (Mac OS)** on a C++ function that has been overridden, the message window appears stating that there are two or more instances.

Stepping Through Messages

When the compiler finds errors during a build, or the CodeWarrior IDE search command finds text you asked it to look for when Using Batch Searches, you'll see the message window.

The window is divided into two panes:

- Message List Pane, which lists the messages, or
- Source Code Pane, which displays the source code for the selected message.

See "Guided Tour of the Message Window" on page 337 for information on the interface items in this window.

To step through the list of messages, click the up or down Stepping Buttons or click the error message you are interested in.

To navigate the source code that is shown in the Source Code Pane for a given message, you use the Interface Pop-Up Menu, Routine Pop-Up Menu, or the Line Number Button. To learn about how to

use these navigational features, refer to "Guided Tour of the Editor Window" on page 130.

Correcting Compiler Errors and Warnings

When an error occurs during compilation, the Message window will show you the error message in the Message List Pane. The location in the source code that the message refers to will be shown in the Source Code Pane. You can navigate to the spot in your source code where the message refers to, and inspect or correct your code.

For a complete list of compiler errors and their possible causes, consult the *Error Reference* documentation on your CodeWarrior CD.

Correcting Errors in the Source Code Pane

To correct a compiler error or warning, you must first find the cause. First, make sure that the Source Code Pane of the Message window is visible. If it isn't visible, refer to "Source Code Disclosure Triangle" on page 340 to learn how to make it visible.

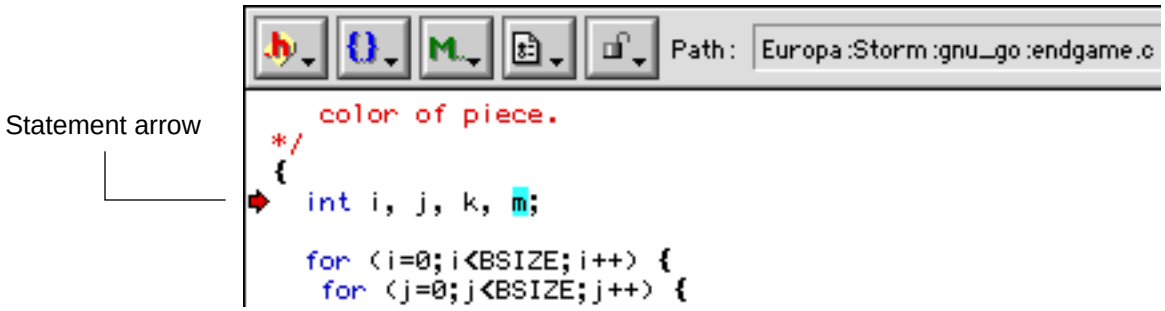
To view the statement that the compiler believes has caused the error or warning, select the message in the Message List Pane of the Message window. Notice that the Source Code Pane view now shows the source code that corresponds to the message. A statement arrow points to the line of code that the compiler reports as an error. The statement arrow is shown in Figure 10.4. Using the statement arrow as a guide, you can then edit the erroneous line directly in the Source Code Pane view.

Use the Interface Pop-Up Menu, Routine Pop-Up Menu, or the Line Number Button in the Source Code Pane to navigate your code or open interface files. To learn about how to use these navigational features, refer to "Guided Tour of the Editor Window" on page 130.

Opening the File for the Corresponding Message

To open a source code file that corresponds to a given message, select the message in the Message List Pane and press Enter/Return. You may also double-click the message in the Message List Pane to open the relevant file.

Figure 10.4 Statement arrow pointing to an error



Correcting Linker Errors

When your project is linked, any errors that may occur can be viewed and corrected.

Viewing Linker Errors

If the linker encounters any errors while linking your project, the Message window appears indicating these errors. This window can be scrolled through by using the scroll bar or Stepping Buttons.

To learn about how to scroll through messages in the Message window, refer to "Stepping Buttons" on page 339. To learn about changing the view of messages in the Message window, refer to "Seeing Errors and Warnings" on page 341.

Since Linker errors are a result of problems in the object code, the CodeWarrior IDE cannot show their corresponding errors in the project's source code files.

Why Linker Errors Occur

Linker errors are usually the result of one of the following circumstances:

- You have misspelled the name of a library routine. This means that the routine that the Linker is searching for does not exist. Check the name of the routine to make sure it is spelled correctly.

- Your compiled code generates a reference too large to be handled by the selected code model.
- Your project is missing the necessary libraries. Linker error messages of this type occur when the project is missing a library. To find out which libraries or shared libraries should be added to your project, refer to the *CodeWarrior Targeting* manual appropriate for your platform, as described in Table 1.2 on page 25.

NOTE: (Mac OS) To find the libraries you should be using, refer to the next section, "Finding Which Library to Use (Mac OS)".

Finding Which Library to Use (Mac OS)

Often a linker error occurs when a library is missing from a project. As a result, one or more functions or identifiers cannot be found and the linker reports a problem. The difficulty you face is trying to figure out which library defines the function or identifier.

To find out which libraries or shared libraries should be added to your project, you can refer to the *CodeWarrior Targeting* manual appropriate for your target, as described in Table 1.2 on page 25. In addition, you can use the **Find Library** tool included on the CodeWarrior Tools CD.

Find Library is a set of files with the internal names of functions and global identifiers from many of the libraries included with the CodeWarrior IDE for Mac OS and ANSI Libraries. Use these files as a file set with the CodeWarrior IDE Find dialog box. Search for the missing identifier in these files, and you can quickly find the right library to include in your project.

Here's how to use the Find Library tool.

1. **Create a File Set**

   Use the Find window to create a set that includes all the files in the `Find Library` folder on your CodeWarrior Tools CD. To learn how to create and use a file set, refer to "Choosing Files to be Searched" on page 183 and "Saving a File Set" on page 187.

2. **Set the search parameters to ignore case.**

   Be sure to select the **Ignore Case** option when doing your search with the Find window. Some identifiers may be all uppercase or lowercase.

3. **Search for the symbol that the linker can't find.**

   Enter the name of the unlinked symbol, and do a batch search of the Find Library file set. When the search has found a matching identifier, you only need to check the file name of the library to determine the name of the library to include in your CodeWarrior project. There is no path information given in the files, so you'll have to locate the correct library in the appropriate CodeWarrior subdirectory on your hard disk.

   For example, if you do a batch search for the identifier `num2dec` in this file set, you'll get a match in the `MathLib` file. You now know that if you want to use the `num2dec` function, you must include the `MathLib` file in your project.

WARNING! The files in the Find Library folder are just compilations of names that have been taken out of the libraries and put into a special naming file. These are *not* the actual library files. Do *not* include the 'Find Library' files in your projects.

NOTE: Some names may be duplicated in more than one file. If the name has a leading period, it is probably a declaration in that file, not a definition. Be sure to use Command-G or batch search to retrieve all files with matching identifiers. Then select the appropriate library for your particular project.

Some of the 68K and PowerPC shared libraries differ slightly, although the library names are the same. To identify the correct library, we've added the suffix -68K or -PPC as appropriate. For example, `DragLib` is listed in the Find Library as both `DragLib -68k` and `DragLib -PPC`. Ignore the suffix when looking for the library by name, but make sure you include the correct CPU version of the library for your target platform.

The CodeWarrior IDE will report the lines on which matches occur, but you can ignore those line numbers. The line information reported is irrelevant with respect to the actual library, and refers only to the line number in the special naming file.

There are several versions of the ANSI Libraries. They all contain the same identifiers, so only one version is included. You should assume that the appropriate library for your project will contain the same functionality.

The ANSI C++ Library file will be difficult to understand because of name mangling. However, most linker errors will list a function name as well as the identifier. Searching for that function name should give you good results. For example, you might get an error message saying "__aad__9bitstringFRC9bitstring in bitsand.c" is unknown. You could either search for the mangled name or you could search for bitsand to get a match.

Correcting Pascal Circular References

The CodeWarrior Pascal compiler's **Make** and **Run** commands build your project by examining every Pascal file in your project file. As this examination is performed, a tree of dependencies is built for the interfaces of your units and for their implementations.

A circular reference occurs when a unit declares something that is used in another unit and that same unit declares something used by the former. To break this loop, the Pascal compiler does not allow such things among the interface parts of units, but it is permitted for implementations.

The example in Listing 10.1 is perfectly valid, since both A's and B's interfaces depend on C's, but are independent from one another. Knowing everything that was declared, A's implementation depends on all interfaces, the same is true for B's and C's. For this example, the make utility will ask the compiler to compile Listing 10.1 in the following order:

Listing 10.1 A valid example of circular referencing

```
UNIT A;          UNIT B;          UNIT C;
INTERFACE        INTERFACE        INTERFACE
USES C;          USES C;

TYPE             TYPE             TYPE
  A_type = ...     B_type = ....    C_type = ...

IMPLEMENTATION   IMPLEMENTATION   IMPLEMENTATION
USES B;          USES A;          USES A, B;
....             ...              ...
```

1. **C's interface is compiled.**
2. **B's interface is compiled.**
3. **All of unit A is compiled (unit and implementation),**
4. **B's implementation is compiled.**
5. **C's implementation is compiled.**

After an interface compilation, the compiler writes a binary symbol table, containing all the declarations of the interface, in a 'sbmf' resource in the project file. This information is read back when the unit's name is encountered in a USES clause for another compilation. A unit is recompiled only when one of the following conditions occur:

- The source was modified,
- The source is currently open and edited, or,
- A unit on which the source depends was recompiled.

Saving and Printing the Message Window

To print or save the contents of the Message window, just follow these steps.

To Print the Message Window:

1. **Make the Message window active.**

   To accomplish this, either click on the deactivated Message window, or select the **Errors & Warnings Window** command from the Window Menu.

2. **Select the Print command from the File Menu.**

   If you choose the **Print** command on the File Menu to print the Message window, the print dialog box appears. Specify printing options and click **OK**. All the errors, warnings, and messages will be printed.

   To learn more about printing, refer to the documentation that came with your printer.

To Save the Message Window:

1. **Make the Message window active.**

   To accomplish this, either click on the deactivated Message window, or select the **Errors & Warnings Window** command from the Window Menu.

2. **Select the Save A Copy As command from the File Menu.**

   The **Save A Copy As** command will display a standard file dialog box.

3. **Specify the name of the file and the location.**

   A text file will be saved containing all the errors, warnings, and messages that are listed in the message window.

Locating Errors in Modified Files

If an error is corrected or the source code is changed, the compiler may not be able to find other errors in the source code file. This may result in an alert telling the user that the position of the error could not be found. When this happens, recompile your project to update the list of errors in the Message window.

Special Library Options (Mac OS)

There are a few special linker options you can use in your project. These options are only accessible from the Project Inspector window as shown in Figure 10.5, available from the Window Menu. These options are only available for library files in the project, and are disabled for text files in the project.

Figure 10.5 Special Library Options

The topics in this section are:

- Import Weak
- Initialize Before
- Merge Into Output

Import Weak

Import Weak tells the operating system to ignore unresolved symbols at load time. Use this option for features not always installed on all machines (such as QuickDraw GX or QuickTime). For more information on shared libraries, see the *Targeting Mac OS* manual.

Initialize Before

Initialize Before indicates which PEF container (shared library or application) gets initialized first. By default, imported PEF containers are initialized before the PEF containers that import them. Use this option for mutually-dependent PEF containers, to specify which gets initialized first.

Merge Into Output

Merge Into Output allows you to put a copy of a shared library into your project's output file. For instance, if you have created your own custom shared library, you can merge it into the application's data fork, guaranteeing that the system finds the custom shared library.

This option may be useful for creating fat libraries as well, where you want to include both CFM68K and PowerPC code.

NOTE: When you merge a shared library into your application, you are copying all the code fragments in that library. If the library contains both 68K and PowerPC code fragments, your application's data file gets them all, which is desirable for fat libraries, but in other cases may increase size unnecessarily.

11

Configuring Version Control Software

This chapter explains how to use the CodeWarrior IDE version control integration facilities to control your source code.

Version Control System Overview

The CodeWarrior IDE includes features for integrating your projects with revision control systems, such as *Metrowerks Visual SourceSafe for Macintosh* (formerly *MW CodeManager*), *Microsoft Visual SourceSafe*, and *MW SourceServer*.

The sections in this chapter are:

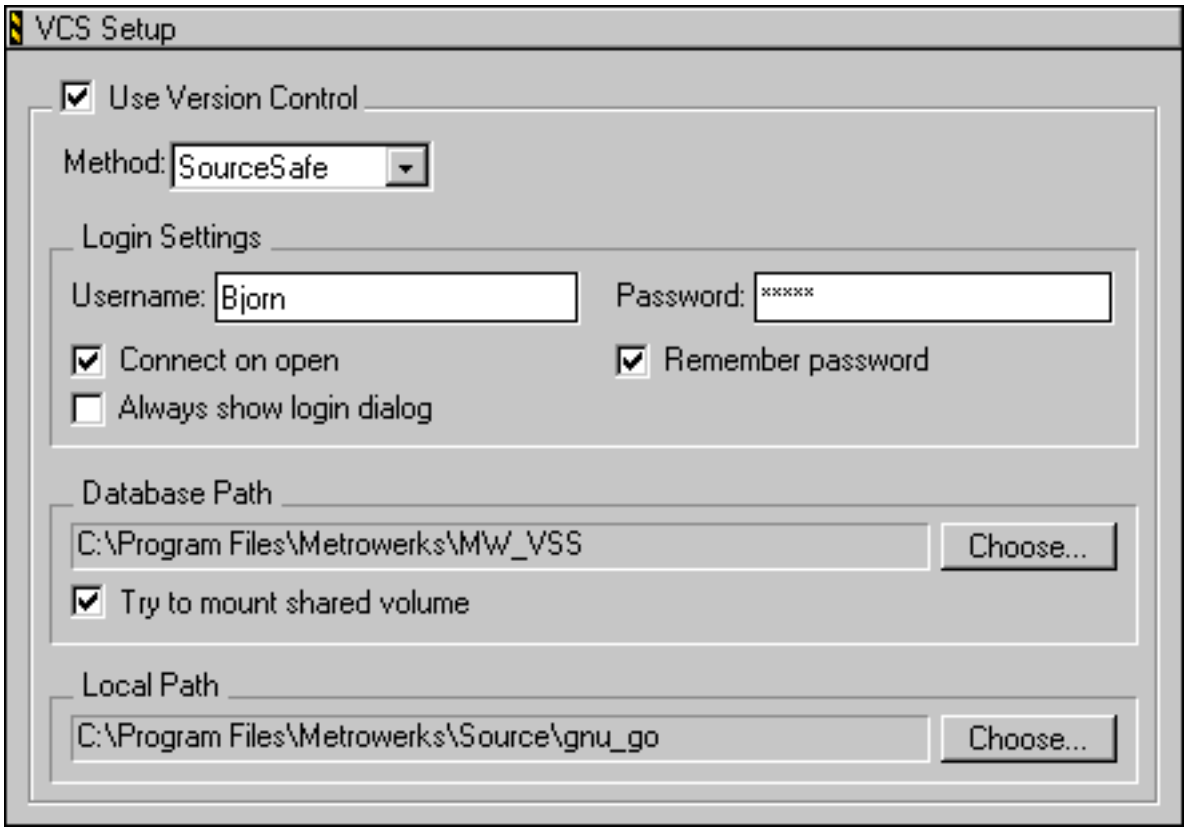
- Version Control System (VCS) Setup
- Setting up a Project for Revision Control
- Using Source Code Control with Files
- Advanced VCS Operations

NOTE: Neither *Metrowerks Visual SourceSafe for Macintosh* nor *Microsoft Visual SourceSafe* is included with CodeWarrior Professional. For more information on either of these products, please contact the respective company.

Version Control System (VCS) Setup

The VCS settings panel, as shown in Figure 11.1, is present if you have installed recognized revision control software for use with the CodeWarrior IDE. Use the VCS Setup panel to specify client options for connecting to a version control system.

Figure 11.1 The VCS settings panel



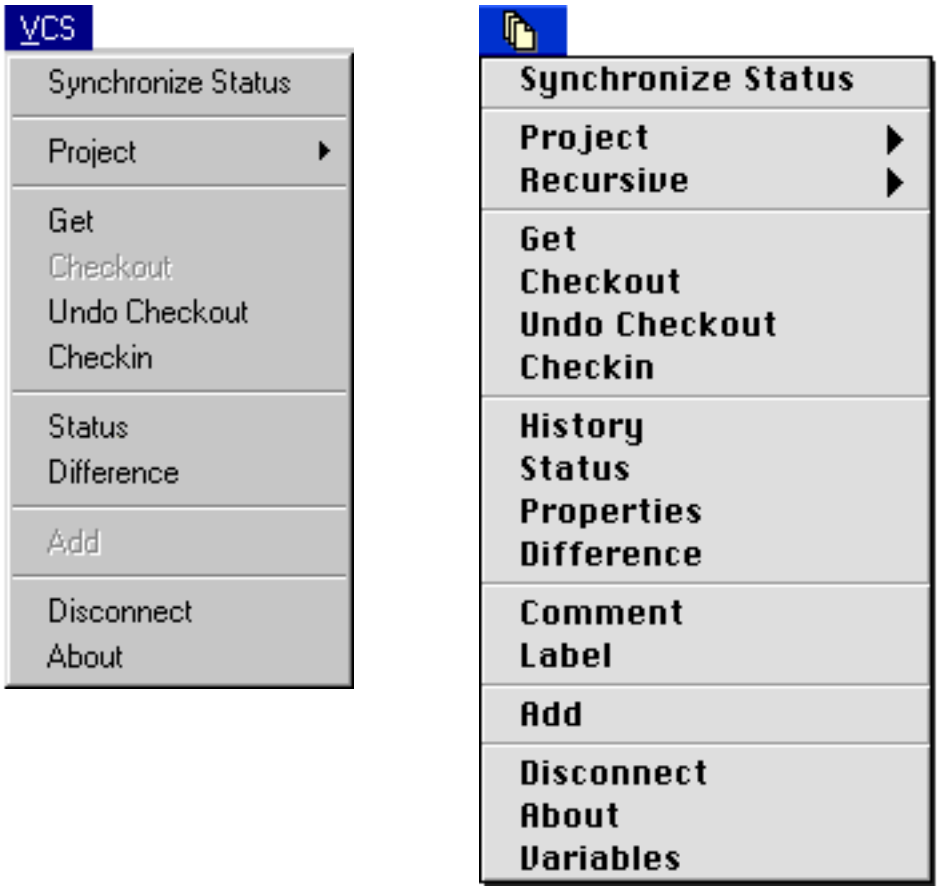
Windows CodeWarrior is capable of connecting to Microsoft Visual SourceSafe on PCs running Windows.

Mac OS Metrowerks Visual SourceSafe for Macintosh is a recognized version control system for the Mac OS.

NOTE: Both *Microsoft Visual SourceSafe* and *Metrowerks Visual SourceSafe for Macintosh* are available separately from their respective manufacturer. Neither product is included with CodeWarrior Professional.

Once **Use Version Control** is enabled, a VCS menu appears on the menubar as shown in VCS menus for Windows and Mac OS. You select commands from this VCS menu to Get, Checkin, Checkout, and Undo Checkout items from the active version control system.

Figure 11.2 VCS menus for Windows and Mac OS



NOTE: The VCS menu differs between the Windows and Mac OS versions of SourceSafe due to the level of support offered by the VCS plug-in.

For more information about using a revision control system with the CodeWarrior IDE, refer to "Version Control System Overview" on page 353. For detailed information on configuration of your revision control software, refer to the documentation that came with the product.

TIP: If you use the Method Pop-up menu in this panel to select "None," the Checkout Status column in the CodeWarrior IDE Project window will display the same states as the CodeWarrior IDE Editor's VCS Pop-Up Menu, checking only file locks and '`ckid`' resources.

Setting up a Project for Revision Control

Setting up a CodeWarrior project for version control is simple. You just need to enter some information in the Version Control System (VCS) Setup panel while a project is open. You'll need to input the information separately for each project that uses revision control.

NOTE: For purposes of this exercise we will use SourceSafe as the version control software. SourceSafe-compatible databases are cross-platform and available in Windows, Mac OS, and UNIX versions.

WARNING! If you change the VCS Settings while you're connected to a SourceSafe database, the CodeWarrior IDE may disconnect you.

SourceSafe must be installed before any of this configuration will work properly.

NOTE: See the documentation that comes with the version of SourceSafe you are using for installation instructions.

TIP: Before starting this exercise, ensure that you already have access to a SourceSafe-compatible database and a user account available. See your database administrator for information on the database's location, your username, password, etc. as you will need that information in order to access the project database.

To Activate Your VCS Settings:

1. **Choose Version Control Settings from the Edit menu.**

   The VCS Settings window appears.

2. **Enable the Use Version Control option.**

   Click the **Use Version Control** option to activate version control.

3. **Choose a version control system from the Method menu.**

   The CodeWarrior IDE is capable of supporting several different types of version control. For this exercise you should specify that you're using SourceSafe.

   NOTE: This option is called SourceSafe since it works with any SourceSafe-compatible database, including MS Visual SourceSafe and Metrowerks Visual SourceSafe for Macintosh.

4. **Enter your username.**

   Enter your SourceSafe user name in the **Username** field.

5. **Enter your password if desired.**

   To avoid the Login dialog, enable the **Remember Password** option and enter your password in the **Password** field.

   If you don't enter your password, the CodeWarrior IDE will display the Login dialog each time you connect to a SourceSafe database.

6. **Enable the Always Show Login Dialog option if desired.**

   If you want to see the Login dialog even though you entered your password, turn on the **Always Show Login Dialog** option. This option is useful if you usually use the same SourceSafe username, but may want to use a different one occasionally.

7. **Enable the Connect on Open option if desired.**

   If you enable the **Connect on Open** option, the CodeWarrior IDE connects you to the SourceSafe database each time you choose a command from the VCS menu. If you leave this option off, you have to connect to the database yourself by choosing the **Connect** command from the VCS menu.

8. **Locate the SourceSafe database folder.**

   Click the **Choose** button and locate the database folder from the open file dialog that appears. Your database folder is where the SourceSafe database is located, and it should be the same directory specified by the `Data_Path` variable.

   The folder's pathname appears in the Database Path field.

9. **Choose a local folder.**

   Click the **Choose** button to set the local path to your working directory. The local working directory folder is where SourceSafe places files after checking them out of the specified project database and where the CodeWarrior IDE expects to find your files.

   The folder's pathname appears in the Local Path field.

NOTE: The SourceSafe options can be used with any SourceSafe-compatible database, including Metrowerks Visual SourceSafe for Macintosh.

---

10. **Click Save.**

    Save your VCS Setting changes using the **Save** button.

To Set a Database Project

1. **Click the SourceSafe panel.**

   Select the SourceSafe panel from the Version Control group in the VCS Settings Panel list. The SourceSafe panel appears.

2. **Enter the project path to use.**

   Enter the path to a project in the database to use as the default project. See your database administrator for this information if you don't already have it.

3. **Click Save.**

   Click the **Save** button to preserve your project setting.

   Congratulations! If all your data was correct, you should be connected to the SourceSafe-compatible database set in the Database Path field. If you aren't, review the above steps, ensuring that the data entered is correct.

Using Source Code Control with Files

A revision control system, such as *Microsoft Visual SourceSafe* or *Metrowerks Visual SourceSafe for Macintosh*, allows you to maintain a database of your source code, and then check files in or out of the database. This makes it easy to track changes to your code, particularly when more than one person is working on your software project.

There are two primary ways to use revision control from the CodeWarrior IDE. You can use the VCS menu (Figure 11.2) that appears on the menubar or the VCS Pop-Up menus (Figure 11.3) that appear in all Editor windows.

Windows A software plug-in for *Microsoft Visual SourceSafe* is included with the CodeWarrior IDE, and revision control operations can be used if you have SourceSafe (available separately) already installed on your system.

NOTE: If you are working on a Macintosh, you can learn more about the VCS menu in the *MW Visual SourceSafe Plug-in Guide.* It is available as part of the *MW Visual SourceSafe for Macintosh* product, purchased separately from the CodeWarrior product.

Mac OS There is a third, more obscure method available on the Macintosh. You can use ToolServer and the command-line MW Visual SourceSafe tools. To learn more about the command-line MW Visual SourceSafe tools, refer to the MW Visual SourceSafe documentation. To learn more about ToolServer, refer to "Using MPW ToolServer Overview" on page 513.

The topics in this section include:

- Determining Version Control Status of a File
- Modifying a Read-Only Source Code File
- Other Revision Control Operations
- Viewing a File's Status in the Project Window
- Mac OS 'ckid' Resources

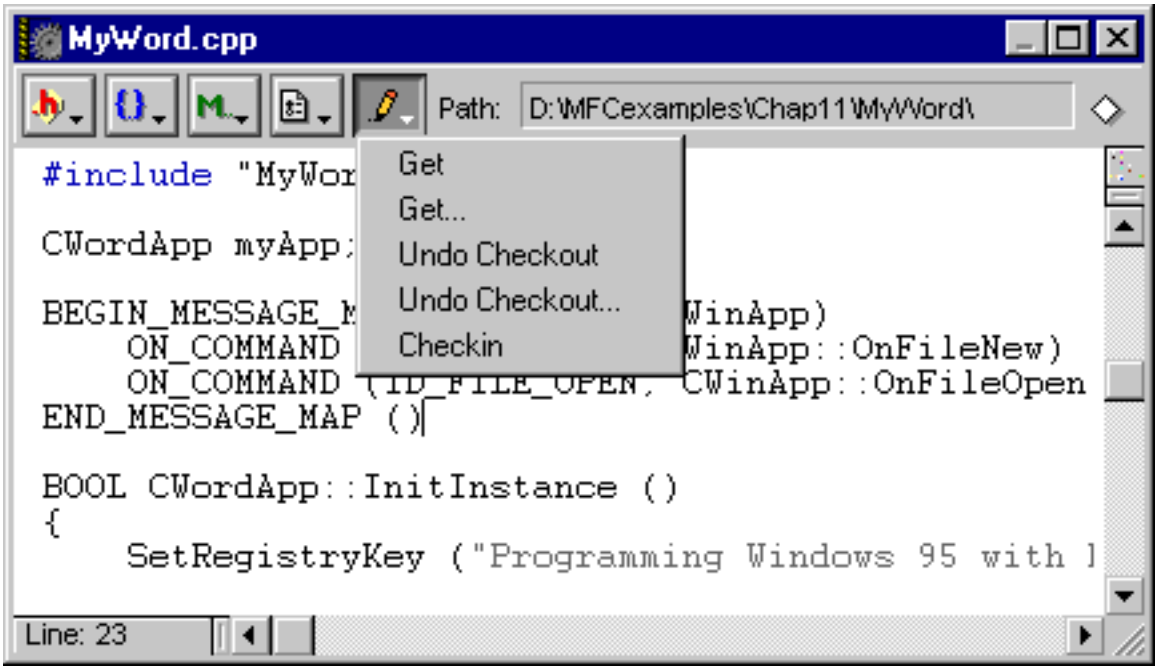
Determining Version Control Status of a File

The CodeWarrior Editor displays the revision control status of the file in the VCS Pop-Up Menu in the Editor window, as shown in Figure 11.3. If version control is not activated or unavailable the pop-up shown in Figure 11.4 appears.

If a file is not already added to the revision control system database, you will see a menu as shown in Figure 11.4.

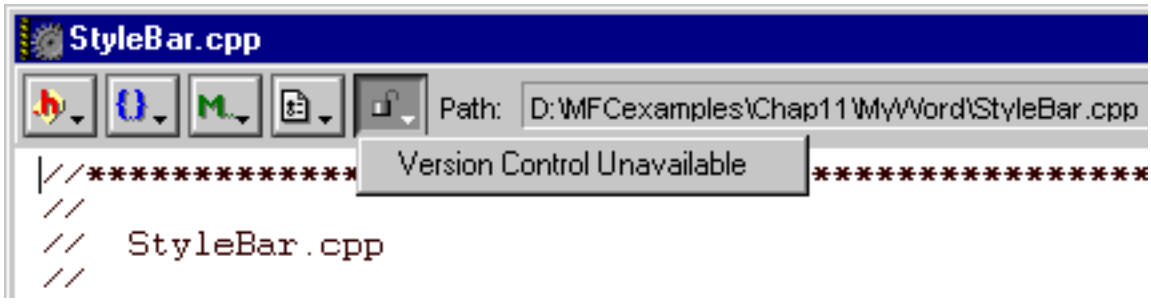
Mac OS The CodeWarrior Editor displays the revision control status of the file, which is controlled by the '`ckid`' resource, in the VCS Pop-Up Menu in the Editor window, as shown in Figure 11.3. Both Projector from Apple and MW Visual SourceSafe use the '`ckid`' resources to mark files that are in revision control with the permissions of the file.

Figure 11.3 VCS Pop-Up Menu



To understand the permissions of a file you are viewing in the Editor, look at the VCS Pop-Up Menu and compare the icon in the menu with the icons shown in Table 11.1.

Figure 11.4 VCS Not Available



TIP: (Mac OS) Some of the icons shown in Table 11.1 can also appear if you are using files that have a 'ckid' resource in them. Many programs can put 'ckid' resources in files to track file per-

missions. One such program that can do this is the Projector tool in MPW, included on your CodeWarrior CD. For more information about 'ckid' resources, refer to the *Building and Managing Programs in MPW* document on your CodeWarrior Reference CD.

Table 11.1 File Permissions Icons

If the icon is…	Then…
Checked out	You can edit the file and add your changes to the revision control database.
Modify Read-Only	You can edit the file, but you cannot add your changes to the revision control database because the file was not properly checked-out for modification.
Read-Only	You cannot edit the file, and it is part of the revision control database.
Unlocked	The file can be edited, and it is not checked into a revision control database.
Locked	You cannot edit the file, and it is *not* part of the revision control database. You may not have the access privileges needed to access the file, or someone may have locked the file.

Modifying a Read-Only Source Code File

You can change the status of a Read-Only source code control file so you can modify it, but after modification, you will not be able to check it back into the source code control database. This feature is provided to allow you to experiment with a file by temporarily making it writable. If the revision control database allowed you to override permissions on files without consequence, it would not be providing any valuable control services for you.

Windows Right-mouse click on the file in Explorer, select **Properties** from the menu that pops up, and see if the **Locked** attribute is checked. If you uncheck it, the file can then be edited.

Mac OS Click the Read-Only icon in the VCS pop-up menu in the Editor window so that the menu pops up, as shown in Figure 11.5. Choose the **Make Writable** command from the menu. The VCS Pop-Up Menu icon changes to Modify Read-Only . You can now modify the file, but you won't be able to check it back into its source code control database.

Figure 11.5 Changing Permissions from Read-Only to Modify Read-Only

Other Revision Control Operations

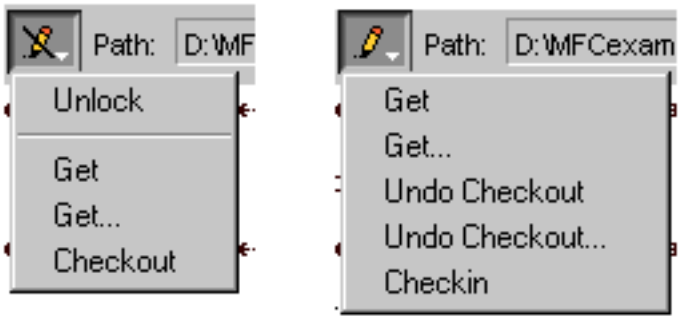
When using the VCS Pop-up Menu, there are some other operations you can perform from the menus.

Windows Refer to Figure 11.6 to see the various menus that occur depending on the revision control status of the file, and the Explorer attributes (such as locked) for the file.

Mac OS Refer to Figure 11.6 to see the various menus that occur depending on the state of the 'ckid' resource and the Finder lock for the file.

TIP: (Mac OS) If for some reason the CodeWarrior IDE gets confused about the VCS status of a file, hold down the Command key and click the VCS Menu to enable all VCS commands, regardless of a file's status. This might be required if you've added a file to the project that already contained a 'ckid' resource.

Figure 11.6 Other Permissions Pop-Up Menu Commands



Depending on the state of the VCS Pop-Up Menu when you click on it, certain operations may be performed on the file you are editing.

- **Unlock**—change the lock on the file (if possible) so that the file is writable
- **Add**—add the file to the revision control database
- **Make Writable** (Mac OS)—make the file writable for experimentation, though it will not be able to be checked into the revision control database
- **Get**—retrieve a fresh copy of the file from the revision control database
- **Checkout**—check the file out from the revision control database for modifications
- **Undo Checkout**—discard any changes made to the file, and tell the revision control database to cancel the checkout of the file.
- **Checkin**—tell the revision control database to accept the file with the changes that have been made to it.

For more information on these operations, refer to the documentation that came with your revision control system.

Windows Note that some items may have an ellipsis character "…" after them. This means that advanced features for this operation are available. To learn more about these operations, refer to "Advanced VCS Operations" on page 366.

Viewing a File's Status in the Project Window

If you have configured your project file to use a revision control system, and have checked in your files in accordance with the procedures described in the revision control system documentation, you will see one of the icons of Table 11.1 in the Checkout Status Column of the Project window for every file under revision control in your project. Refer to Figure 11.7 to see where the Checkout Status Column is located in the Project window.

Clicking the Checkout Status Column icon at the top of the column will perform a **Synchronize Status** command on the project's files against the revision control database. The status of all files you have on your hard disk will be compared and synchronized with the status of the files in the revision control database.

Mac OS For more information about the **Synchronize Status** command, refer to the *MW Visual SourceSafe for Macintosh* documentation.

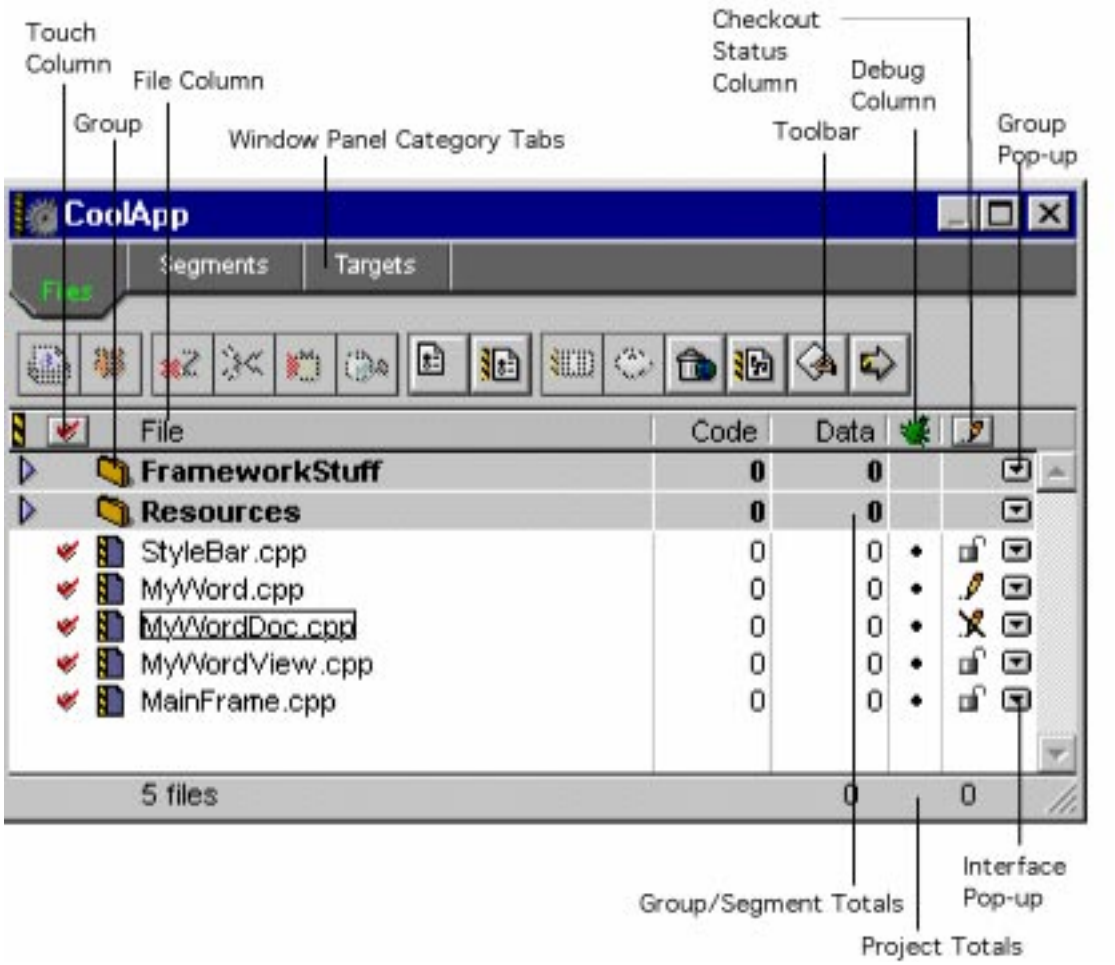
Mac OS 'ckid' Resources

The CodeWarrior IDE *almost* always respects the presence of 'ckid' resources in text files. The 'ckid' resources are used by Projector and MW Visual SourceSafe to track whether a file is writable or not, and are stored within the resource fork of your source code files.

The one time that the CodeWarrior IDE may do something different than indicated by a 'ckid' resource is when dealing with a project file that contains a 'ckid' resource.

To learn more about how to view the checkout status of a Project window, refer to "Project Checkout Status icon" on page 48.

Figure 11.7 Checkout Status Column



Advanced VCS Operations

To access more advanced VCS features, choose the optional form of the menu command. You'll notice that the some of the menu commands have an ellipsis (…) next to them, indicating that more options are available when you select the command.

- Windows VCS Features
- Mac OS VCS Features
- VCS Window

NOTE: For purposes of these instructions, we will use MS Visual SourceSafe as the source-control program. Other source-control programs may have their own unique windows. See their documentation for details.

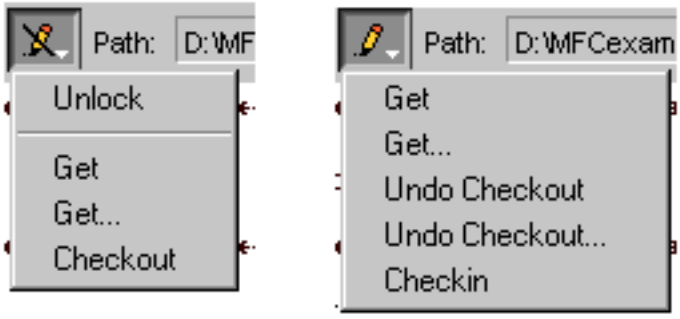
Windows VCS Features

The features present in your source-control system are entirely dependent upon that particular product.

Windows VCS Commands

Choose commands from the VCS menu that contain the ellipsis (…) character after them, as shown in Figure 11.8.

Figure 11.8 VCS Commands (Windows)



Windows Supported Commands

The commands shown in Table 11.2 are fully supported from within the CodeWarrior IDE.

Table 11.2 Supported Commands

Command	Description
Synchronize Status	Updates the VCS Status column in the Project window by examining each file's status and updating its display information.
Project	Contains a submenu of VCS commands that enable you to perform Get, Checkout, Undo Checkout, Checkin, Status, and Add operations on entire CodeWarrior project files.
Get	Retrieves a copy of the file without checking it out of the project database.
Checkout	Checkout files for exclusive modification.
Undo Checkout	Cancels a checkout, voiding all changes.
Checkin	Returns a modified file to a project and relinquishes exclusive control.
Status	Displays a file's checkout status.
Add	Adds a file to the current project.
Connect or Disconnect	Connects or disconnects you from the project database depending upon the current open status of the project database.
About	Displays copyright and version information.

Connect

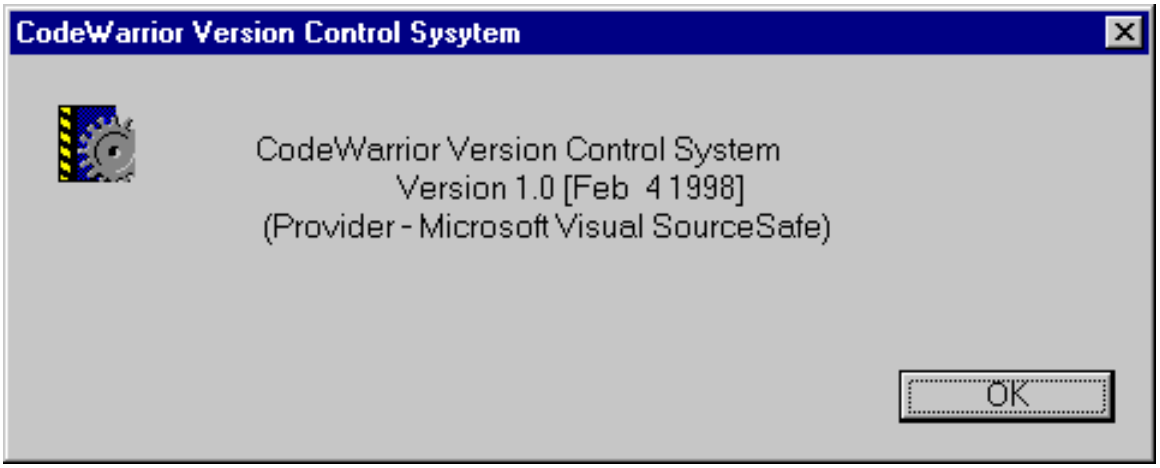
When the **Connect** command is selected and **Always Show Logon** is enabled, the window shown in Figure 11.9 appears.

Figure 11.9 VCS Connect window

About

To determine which version of the CodeWarrior VCS you are running, choose **About** from the VCS menu to view the dialog shown in Figure 11.10.

Figure 11.10 VCS About window



Mac OS VCS Features

Mac OS To view ellipsis characters in a command menu on the Macintosh, press the Option key while clicking on the VCS Pop-Up Menu, as shown in Figure 11.11.

Figure 11.11 Advanced VCS Features (Mac OS)

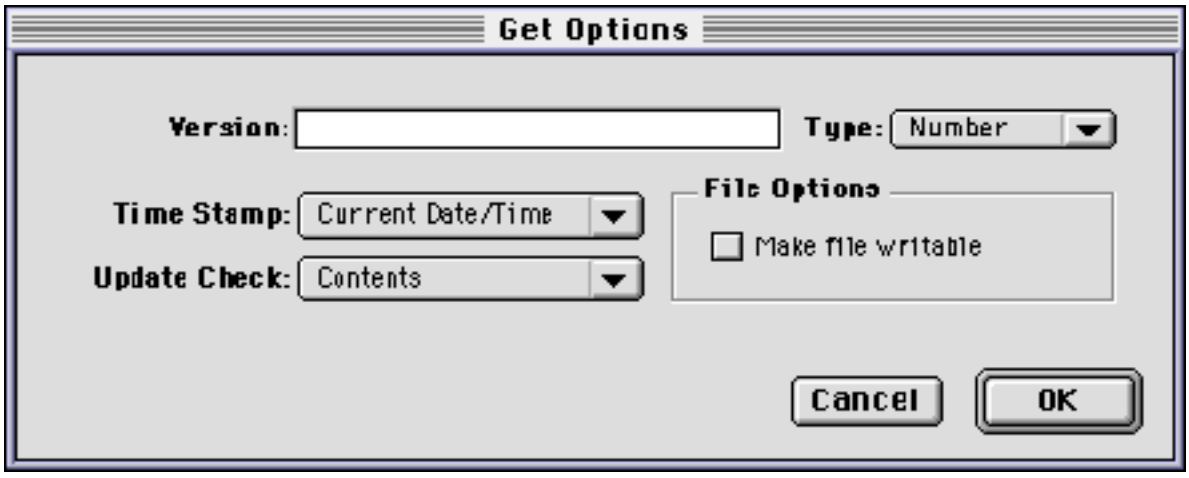
To learn about each of the operations in these dialogs, refer to the *MW Visual SourceSafe for Macintosh* documentation.

- Checkin Options—shown in Figure 11.12
- Checkout Options—shown in Figure 11.13
- Get Options—shown in Figure 11.14

Figure 11.12 Advanced VCS Checkin Options

Figure 11.13 Advanced VCS Checkout Options

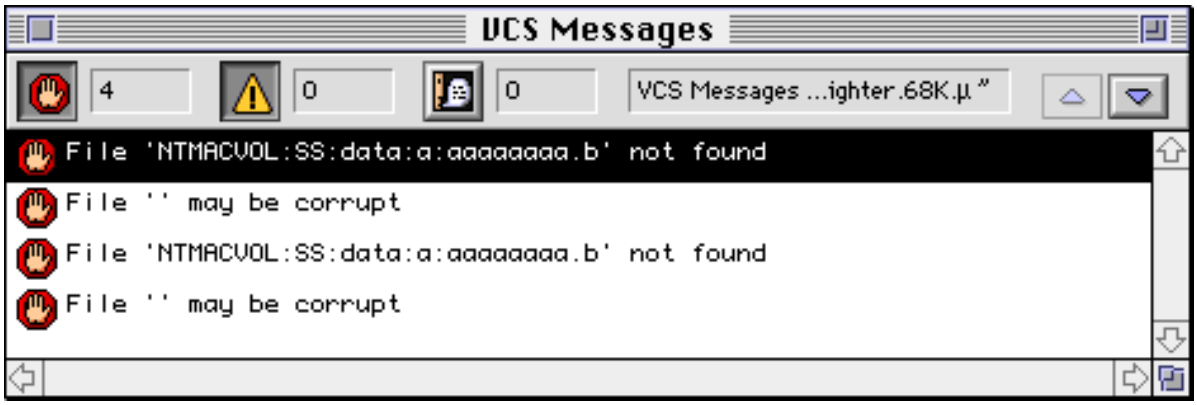
Figure 11.14 Advanced VCS Get Options



VCS Window

The CodeWarrior IDE now shows a variant of the Message Window for revision control messages, as shown in Figure 11.15. To learn how to use the controls in the window, refer to "Using the Message Window" on page 340.

Figure 11.15 VCS Window



12

IDE Menu Reference

This chapter describes each command on each CodeWarrior IDE menu.

IDE Menu Reference Overview

There are several menus in the CodeWarrior IDE menu bar:

- File Menu
- Edit Menu
- Search Menu
- Project Menu
- Debug Menu
- Data Menu
- Window Menu
- Version Control System (VCS) Menu
- VCS menus for Windows and Mac OS
- Toolbar Submenu
- Java Exceptions Submenu
- Apple Menu (Mac OS)
- Tools Menu (Mac OS)
- Scripts Menu (Mac OS)
- Editor Extensions Menu (Mac OS)
- Info Menu (Solaris)

Windows Under Windows, the File Menu, Edit Menu, Search Menu, Project Menu, Debug Menu, Window Menu, and VCS menus for Windows and Mac OS are visible at all times.

Mac OS Under the Mac OS, the Apple Menu (Mac OS), File Menu, Edit Menu, Search Menu, Project Menu, Debug Menu, Window Menu, and VCS menus for Windows and Mac OS are visible at all times. If you activate ToolServer via the IDE Extras preference panel, then the Tools Menu (Mac OS) is also visible at all times.

The Version Control System (VCS) Menu appears only if you have installed and configured your CodeWarrior product to work with a compatible revision control system that you purchased separately. To learn more about revision control systems, and how to use them with CodeWarrior, refer to the documentation that came with the additional revision control software.

The Data Menu appears when a window that can use its commands is present on the screen.

Mac OS Only Menus

The Tools Menu (Mac OS) appears if ToolServer is active. The Scripts Menu (Mac OS) and To learn more about how to configure the appropriate preferences so this menu becomes visible, refer to "IDE Extras Panel" on page 242. only appear when you've set the correct preferences in the Preferences window. Refer to "IDE Extras Panel" on page 242 for more information on how to do this.

In addition, you can make the commands that appear in many menus change by holding down the Option or Shift keys when pulling the menu down.

The text of some menu commands may change depending on the context of the action, or actions, performed.

Many menu commands can also have button equivalents on the toolbars. To learn more about how to customize the toolbars, refer to "Customizing Toolbars" on page 280.

File Menu

The File Menu contains commands you use when opening, creating, saving, closing, and printing existing or new source code files and

projects. The File Menu also provides a few different methods of saving edited files.

New

Creates a new editable text file.

To learn more about this command, refer to "Creating a New File" on page 101 for more information.

New Project

Creates a new project file.

To learn more about this command, refer to "Working with Project Stationery" on page 56.

New Empty Project

Creates a new project file without using stationery.

Windows This command is available by pressing the key binding Ctrl-Alt-Shift-N.

Mac OS This command is available when you press the Option key while clicking on the File menu.

To learn more about this command, refer to "Creating a New Project" on page 50.

Open

Allows you to open an existing text file.

To learn more about this command, refer to "Opening Files from the File Menu" on page 102.

Open Recent

The **Open Recent** command exposes a submenu of projects and files that have recently been opened. You may choose a file from the sub-menu to instantly open one of these items.

If some files have identical names, the full paths to those files are displayed in order to identify them.

To learn more about this command, refer to "Opening Files from the File Menu" on page 102.

Find and Open File

Opens an existing file, searching the current access paths as specified in the Access Paths panel of the Target Settings window.

See "Opening Files from an Editor Window" on page 107 for more information.

Find and Open 'Filename'

Opens an existing text file, using the currently selected text in the editor window as the target file name.

See "Opening Files from an Editor Window" on page 107 for more information.

Close

Closes the active window whether it is the Project window, the Message Window, or a source code window.

See "Closing a File" on page 115 for more information.

To learn how to close all open Editor windows, refer to "Closing All Files" on page 117.

Close All

Closes all open Editor windows.

Windows This command is available by pressing the key binding Ctrl-Shift-W while closing an active editor window.

Mac OS This command is available when you press the Option key while clicking on the File menu.

To learn more about this topic, refer to "Closing All Files" on page 117.

Switch to MW Debugger (Mac OS)

Gives control to the CodeWarrior external debugger. The line containing the text insertion point in the CodeWarrior Editor is displayed by the debugger. This command is dimmed if no source code window is active, or the debugger is not running.

To learn more about this topic, refer to the *CodeWarrior Debugger User Guide*.

Save

Saves the contents of the active Editor window to disk.

For more information on this topic, refer to "Saving one file" on page 109.

Save All

Saves all Editor files that are currently open.

Windows This command is available by pressing the key binding Ctrl-Shift-S.

Mac OS This command is available when you press the Option key while clicking on the File menu.

For more information on this topic, refer to "Saving all files" on page 109.

Save As

Saves the contents of the active window to disk under another name of your choosing.

For more information, see "Renaming and saving a file" on page 110.

Save A Copy As

Saves the active Editor window, Message window, or Project window in a separate file. This command operates in two different ways, depending on whether a source code file or the Project window is active.

For more information, see "Backing up files" on page 111.

Revert

Use the **Revert** command to revert the active Editor window to its last saved version.

To learn more about how to revert to the previous version of a file, see "Reverting to a Previously-Saved File" on page 119.

Print Setup (Windows)

Page Setup (Mac OS)

Sets the options used when printing files from the CodeWarrior IDE.

For more information about this command, see "Setting Print Options" on page 117.

Print

Prints files from the CodeWarrior IDE on your printer.

For more information on printing files, see "Printing a Window" on page 118, or read the documentation that came with your printer.

Exit (Windows)

Quit (Mac OS)

Quits the CodeWarrior IDE immediately, provided one of the following conditions has been met:

- All changes to the open Editor files have already been saved, or

- The open Editor files have not been changed.

If a Project window is open, all changes to the project file are saved before the environment quits. If an Editor window is open and changes have not been saved, the CodeWarrior IDE asks if you want to save the changes before quitting.

Edit Menu

The Edit menu contains all the customary editing commands, along with some CodeWarrior additions. This menu also includes the commands that open the Preferences and Target Settings windows.

Undo

The text of this menu command varies depending on the most recent action, and your Editor options settings.

Undo reverses the effect of your last action. The name of the Undo command varies depending on the type of operation you last executed. For example, if you have just typed in an open Editor window, the Undo command is renamed **Undo Typing**. Choosing the **Undo Typing** command will remove the text you have just typed.

To learn more about this topic, refer to "Undoing the last edit" on page 148, and "Undoing and redoing multiple edits" on page 149.

If you don't have **Use Multiple Undo** turned on in the Editor Settings preference panel, the Undo command toggles between Undo and Redo. To learn more about how to configure this option, refer to "Editor Settings" on page 261.

Redo, Multiple Undo, and Multiple Redo

Once an operation has been undone, it may be redone. For example, if you select the **Undo Typing** command, the command is changed to **Redo Typing**. Choosing this command overrides the previous undo.

If you have **Use Multiple Undo** turned on in the Editor Settings preference panel, you have more flexibility with regard to Undo

and Redo operations. Choose **Undo** multiple times to undo multiple actions. Choose **Redo** multiple times to redo multiple actions.

To learn more about undo operations, refer to "Undoing the last edit" on page 148, and "Undoing and redoing multiple edits" on page 149.

To learn about how to configure multiple undo, refer to "Editor Settings" on page 261.

Cut

Deletes the selected text and puts it in the system Clipboard, replacing the contents of the Clipboard.

Copy

Copies the selected text in the active Editor window onto the system Clipboard. If the Message Window is active, the Copy command copies all the text in the Message Window onto the Clipboard.

Paste

Pastes the contents of the system Clipboard into the active Editor window.

The **Paste** command replaces the selected text with the contents of the Clipboard. If no text is selected, the Clipboard contents are placed after the text insertion point.

If the active window is the Message Window, the **Paste** command is dimmed and cannot be executed.

Clear

Deletes the selected text without placing it in the system Clipboard. The **Clear** command is equivalent to pressing the Delete or Backspace key.

Select All

Selects all the text in the active window. This command is usually used in conjunction with other **Edit** menu commands such as **Cut**, **Copy**, and **Clear**.

To learn more about selecting text, refer to "Selecting Text" on page 144.

Balance

Selects the text enclosed in either parentheses (), brackets [], or braces {}. For a complete procedure on how to use this command and how to balance while typing, consult "Balancing Punctuation" on page 147.

Shift Left

Shifts the selected source code one tab size to the left. The tab size is specified in the Preferences window.

To learn more about this feature, refer to "Shifting Text Left and Right" on page 148.

Shift Right

Shifts the selected source code one tab size to the right.

To learn more about this feature, refer to "Shifting Text Left and Right" on page 148.

Insert Reference Template (Mac OS)

Inserts a routine template corresponding to the selected Mac OS Toolbox call in the active window. The CodeWarrior IDE uses the online reference database application selected in the **Find Reference Using** pop-up menu to search for the routine's definition.

To learn about configuring the online reference database application and the **Find Reference Using** pop-up menu, refer to "IDE Extras Panel" on page 242.

Preferences

Use this command to change the global preferences for the CodeWarrior IDE.

To learn more about configuring preferences, refer to "Choosing Preferences" on page 238.

Target Settings

Use this command to display the Target Settings window where you can change settings for the active build target. Note that the name of this menu command will vary depending on the name of your current build target.

To learn more about the Settings window, refer to "Choosing Target Settings" on page 293. To learn how to change the current build target, refer to "Set Current Target" on page 395.

Version Control Settings

This menu command displays the Version Control System options panel. To learn more about this panel, refer to the section of this manual entitled "Version Control System Overview" on page 353.

If this command is not enabled, you do not have a revision control system configured for use with the CodeWarrior IDE.

Search Menu

The Search Menu contains all the necessary commands used to find text, replace text, and compare files. There are also some commands for code navigation.

Mac OS Use the commands in this menu to find the definitions of routines in your source code and in a library reference database like THINK Reference™ or Apple Macintosh Programmer's Toolbox Assistant™.

Find

Opens the Find dialog box which is used to find and/or replace the occurrences of a specific string in one or many files.

To learn more about the Find window and its capabilities, refer to "Guided Tour of the Find Dialog Box" on page 161.

Find Next

Finds the next occurrence of the Find text box string in the active window. This is an alternative to clicking the **Find** button in the Find dialog box.

To learn more about this feature, refer to "Finding Search Text" on page 175.

Find Previous

Find Previous operates the same way as **Find Next**, except that it finds the *previous* occurrence of the Find text box string.

Windows This command is available by pressing the key binding Shift-F3.

Mac OS Hold down the Shift key while clicking on the Search menu to change the **Find Next** command to **Find Previous**.

To learn more about this feature, refer to "Finding Search Text" on page 175.

Find in Next File

Finds the next occurrence of the Find text box string in the next file listed in the Multi-File Search portion of the Find window (as exposed by the Multi-File Search Disclosure triangle in the Find window). This is an alternative to using the Find window. If the Multi-File Search button is not enabled as shown in Figure 6.3 on page 167, this command is dimmed.

To learn more about this feature, refer to "Finding and Replacing Text in Multiple Files" on page 181.

Find in Previous File

This command operates in much the same way as **Find in Next File**. The **Find in Previous File** command begins at the end of the previous file in the file list and searches for the next occurrence of the Find text box string.

Windows This command is available by pressing the key binding Ctrl-Shift-T.

Mac OS If you hold down the Shift key while clicking on the Search menu, the **Find in Next File** command is changed to **Find in Previous File**.

To learn more about this feature, refer to "Finding and Replacing Text in Multiple Files" on page 181.

Enter 'Find' String

This command copies the selected text in the active window into the Find text box, making it the search target string. This is an alternative to copying text and pasting it into the Find window.

To learn how to select text, refer to "Selecting Text" on page 144.

Enter 'Replace' String

This command copies the selected text in the active window into the Replace text box, making it the replacement string. This is an alternative to selecting the string and copying it into the Find window.

Windows This command is available by pressing the key binding Ctrl-Shift-E.

Mac OS Hold the Shift key down while clicking on the Search Menu to change **Enter 'Find' String** to **Enter 'Replace' String**.

To learn more about replacing text, refer to "Replacing Found Text" on page 178.

Find Selection

Finds the next occurrence of the selected text in the active text editor window.

To learn more about this feature, refer to "Finding Search Text" on page 175.

Find Previous Selection

This command finds the previous occurrence of the selected text in the active text editor window.

Windows This command is available by pressing the key binding Ctrl-Shift-F3.

Mac OS If you hold down the Shift key while clicking on the Search menu, the **Find Selection** command becomes **Find Previous Selection**.

To learn more about this feature, refer to "Finding Search Text" on page 175.

To learn how to select text, refer to "Selecting Text" on page 144.

Replace

This command replaces the selected text in the active window with the text string in the Replace text box of the Find window. If no text is selected in the active editor window, this command is dimmed.

This command is useful if you wish to replace one instance of a text string without having to open the Find window. For example, say that you have just replaced all the occurrences of the variable "`icount`" with "`jcount`". While scrolling through your source code, you notice one instance of the variable "`icount`" is misspelled as "`icont`". To replace this variable with "`jcount`", select "`icont`" and choose the **Replace** command from the Search Menu.

To learn more about replacing text, refer to "Replacing Found Text" on page 178.

To learn how to select text, refer to "Selecting Text" on page 144.

Replace & Find Next

This command replaces the selected text with the string in the Replace text box of the Find window, and then performs a **Find Next**.

If no text is selected in the active editor window and there is no text in the Find text box string field of the Find window, this command is dimmed.

To learn more about replacing text, refer to "Replacing Found Text" on page 178.

To learn how to select text, refer to "Selecting Text" on page 144.

Replace & Find Previous

This command operates the same way as **Replace & Find Next** except that it performs a **Find Previous** after replacing text.

Windows Press the key binding Ctrl-Shift-L to use this command.

Mac OS Hold down the Shift key while clicking on the Search menu to change **Replace & Find Next** to **Replace & Find Previous**.

Replace All

Finds all the occurrences of the Find string and replaces them with the Replace string. If no text is selected in the active editor window and there is no text in the Find text box in the Find dialog box, this command is dimmed.

Find Definition

This command searches for the definition of the routine name selected in the active window. Searching occurs in the source files belonging to the open project. If the definition is found, the CodeWarrior IDE opens the source code file where the routine is defined and highlights the routine name.

If the CodeWarrior IDE finds more than one definition, a Message window appears warning you of multiple definitions. For more information on the Message window, consult "Using the Message Window" on page 340.

If no definition is found, a system beep sounds.

Find Reference (Mac OS)

This command searches for the definition of the routine name selected in the active editor window using the online help system specified in the IDE Extras preference panel.

Searching takes place in QuickHelp™, QuickView™, THINK Reference™ version 2.x, or Toolbox Assistant™.

If no definition is found, a system beep sounds.

For more information about choosing an online reference database, refer to "IDE Extras Panel" on page 242.

To learn more about online databases, refer to "Online References" on page 155.

Find Definition & Reference (Mac OS)

This command searches for the definition of the routine name selected in the active editor window. Searching starts within the source code files belonging to the open project.

To use this command, first select a routine name. Then hold down the Option key and choose **Find Definition & Reference** from the Search Menu.

If the routine definition is not found within the project files, searching continues in QuickHelp™, QuickView™, THINK Reference™ version 2.x, or Toolbox Assistant™.

If no definition is found, a system beep sounds.

For more information about choosing an online reference database, refer to "IDE Extras Panel" on page 242.

To learn more about online databases, refer to "Online References" on page 155.

Go Back

This command returns you to the previous view in the Browser.

To learn more about this feature, refer to "Go Back and Go Forward" on page 227.

Go Forward

This command moves you to the next view in the Browser (after you have used the **Go Back** command to return to a previous view).

Refer to "Go Back and Go Forward" on page 227 for more information.

Go To Line

Opens a dialog box (in which you enter a line number) and then moves the text insertion point to the line number you specify.

For more information about this feature, refer to "Going to a Particular Line" on page 154.

Compare Files

Opens a dialog box to choose two files or folders to compare and merge. After choosing files to compare, a file comparison window appears, showing differences between the two files. If two folders are compared, the differences between the folders are shown in the Compare Folders window. For more information, see "Comparing and Merging Files & Folders" on page 120.

Apply Difference

Adds, removes, or changes text in the destination file shown in a file comparison window that is different from the text in the comparison window's source file.

Unapply Difference

Reverses the action of an **Apply Difference** command in a file comparison window.

Project Menu

The Project menu lets you add and remove files and libraries from your project. It also lets you compile, build, and link your project. All of these commands are covered in this section.

Add Window

This command adds the file in the active Editor window to the open project.

To learn more about this feature, refer to "Using the Add Window command" on page 78.

Add Files

This menu command adds files to the Project window.

To learn more about this feature, refer to "Using the Add Files command" on page 73.

Create New Group

The **Create New Group** command allows you to create a new group in the current project. This command is present in the Project menu if the Files category is selected in the current project window.

For more information about creating groups, refer to "Creating Groups" on page 80.

Create New Target

The **Create New Target** command allows you to create a new build target for the current project. This command is present in the Project menu if the Targets category is selected in the current project window.

For more information about creating targets, refer to "Working with Complex Projects" on page 85.

Create New Segment (Mac OS)

The **Create New Segment** command allows you to create a new segment (also referred to as a group of files) in the current Mac OS 68K project. This command is in the Project menu if the Segments category is selected in the current project window.

For more information about managing segments, refer to "Managing Files in a Project" on page 68.

Remove Selected Items

This menu command removes the currently selected items from the Project window.

To learn more about removing items from the Project window, refer to "Managing Files in a Project" on page 68.

WARNING! This command cannot be undone.

Check Syntax

This command checks the syntax of the source code file in the active Editor window or the selected file(s) in the open Project window. If the active Editor window is empty, or no project is open, this command is dimmed.

Check Syntax does not generate object code. This command only checks the source code for syntax errors. The progress of this operation is tracked in the Toolbar's message area.

To abort this command at any time, press Esc (Windows) or Command-Period (Mac OS).

If one or more errors are detected, the Message window appears. For information on how to correct compiler errors, consult "Correcting Compiler Errors and Warnings" on page 343.

Preprocess

This command performs preprocessing on selected source code files in any language that has a preprocessor, including C, C++, and Pascal.

To learn more about this command, refer to "Preprocessing Source Code" on page 336.

Precompile

This command precompiles the text file in the active Editor window into a precompiled header file.

To learn more about this topic, refer to "Using Precompiled or Preprocessed Headers" on page 329.

Compile

This command compiles selected files. If the project window is active, the selected files and segments/groups are compiled. If a source code file in an Editor window is active, the source code file is compiled. The source code file must be in the open project.

To learn more about this topic, refer to "Compiling and Linking a Project" on page 319.

Disassemble

This command disassembles the compiled source code files selected in the project window, and displays object code in new windows with the title of the source code file and the extension `.dump`.

To learn more about this feature, refer to "Disassembling Source Code" on page 336.

Bring Up To Date

This command updates the open project by compiling all of its modified and "touched" files.

To learn more about this topic, refer to "Updating a Project" on page 322.

Make

This command builds the selected project by compiling and linking the modified and "touched" files in the open project. The results of a successful build depend on the selected project type.

To learn more about this topic, refer to "Making a Project" on page 323.

Remove Object Code

This command removes all compiled source code binaries from the open project. The numbers in the Code column and Data column of each file are reset to zero.

To learn more about this topic, refer to "Removing Objects" on page 327.

Remove Object Code & Compact

This command removes all binaries from the project and compacts it. Compacting the project removes all binary and debugging information and retains only the information regarding the files that belong to the project and project settings.

Windows This command is available by pressing the Ctrl, Shift, and – keys simultaneously.

Mac OS Hold down the Option key while clicking on the Project menu to change the **Remove Object Code** command to **Remove Object Code & Compact**. The **Remove Object Code & Compact** command is also available by pressing the Option, Command, and – keys simultaneously.

To learn more about this topic, refer to "Removing Objects" on page 327.

Re-search for Files

To speed up builds and other project operations, the IDE caches the locations of project files after it has found them in the access paths. **Re-search for Files** forces the IDE to forget the cached locations of files and re-search for them in the access paths. This command is useful if you have moved files around on disk and want the IDE to find them in their new locations.

If the **Save Project Entries Using Relative Paths** setting is enabled the IDE does not reset the relative path information stored with each project entry, so re-searching for files will find the source files in the same location (the exception is if the file no longer exists in the old location). In this case the IDE will only re-search for header files. To force the IDE to also re-search for source files, you must first select **Reset Project Entry Paths**.

If the **Save Project Entries Using Relative Paths** setting is disabled, the IDE will re-search for both header and source files.

Reset Project Entry Paths

This command resets the location information stored with each project entry when the **Save Project Entries Using Relative Paths** setting is enabled. The next time the project entries are accessed, the IDE will re-search for the project entries in the access paths. This command does nothing if the **Save Project Entries Using Relative Paths** setting is disabled.

Synchronize Modification Dates

This command updates the modification dates stored in the project file. It checks the modification date for each file in the project, and if the file has been modified since it was last compiled, the CodeWarrior IDE marks it for recompilation.

To learn more about this topic, refer to "Synchronizing modification dates" on page 84.

Enable Debugger

Disable Debugger

Use these commands to change allowances for project debugging. The **Enable Debugger** command sets preferences to allow your project to be debugged. The **Disable Debugger** sets preferences so no debugging can occur.

Mac OS When you choose the **Enable Debugger** command, the **Run** command changes to **Debug** and lets the debugger launch and debug your project. When you choose the **Disable Debugger** command, the **Run** command runs your project normally.

To learn more about this topic, refer to "Controlling Debugging in a Project" on page 97.

Run

This command compiles, links, creates a stand-alone application, and launches that application.

Windows If the project type is set as a library or a shared library, then the **Run** command is dimmed.

Mac OS If the project type is set as a code resource, library, MPW Tool, or shared library, then the **Run** command is dimmed. Press the Option key to change this command to **Debug**.

To learn more about this topic, refer to "Running a Project" on page 324.

Resume (Mac OS)

This command appears in the **Project** menu when the application created by the CodeWarrior IDE is running already. You choose this command to switch into the application from the CodeWarrior IDE.

Debug

This menu command compiles and links your project and then opens the project's debugger file in the integrated debugger. This

command runs the integrated debugger for any project that the debugger can work with.

Mac OS This command appears in the **Project** menu when you press the Option key to change the **Run** command to **Debug**, or when the debugger is enabled.

To learn more about the integrated debugger, refer to the *CodeWarrior Debugger User Guide*.

Set Default Project

This menu command selects which project is the default project. To learn more about what a default project is, refer to "Choosing a Default Project" on page 67.

Set Current Target

This menu command allows you to choose a different target within the current project to work with. This menu command might be useful if you want to switch between multiple targets in a project and do a build for each one.

Debug Menu

The Debug menu contains commands that allow you to manage program execution. All of these commands are available in both the integrated and external debuggers. There is also a submenu for customizing breaks for Java exceptions.

For more information see the *Debugger User Guide*.

Kill

Permanently terminates execution of the target program and returns control to the IDE.

Reset (Embedded)

Resets the program and returns control to the IDE.

Hard Reset (Embedded)

Resets the program and also resets the hardware before returning control to the IDE.

Step Over

Executes a single statement, stepping over function calls.

Step Into

Executes a single statement, stepping into function calls.

Step Out

Executes the remainder of the current function until it exits to its caller.

Stop

Temporarily suspends execution of the target program and returns control to the integrated debugger.

Set Breakpoint

Clear Breakpoint

Sets or clears a breakpoint in the line in which the text insertion cursor is currently located. This command toggles between **Set Breakpoint** and **Clear Breakpoint**, depending on whether you have a breakpoint currently set in the line.

If you have your editor window set up to show breakpoints (by choosing the **Show Breakpoints** command from the Debug menu), you will see the breakpoints column on the left side of the editor window. If you set a breakpoint in a certain line of code, a marker appears in the breakpoints column next to that line. If you clear an existing breakpoint from a certain line, the marker to the left of that line disappears from the breakpoints column.

Enable Breakpoint

Disable Breakpoint

Enables or disables the breakpoint in the line in which the text insertion cursor is currently located. This command toggles between **Enable Breakpoint** and **Disable Breakpoint**, depending on whether you have a breakpoint currently enabled in the line.

If you have your editor window set up to show breakpoints (by choosing the **Show Breakpoints** command from the Debug menu), you will see the breakpoints column on the left side of the editor window. Darkened markers in the breakpoints column indicate enabled breakpoints. Dimmed markers indicate disabled breakpoints.

Clear All Breakpoints

Clears all breakpoints in all source-code files belonging to the target program.

Show Breakpoints

Hide Breakpoints

Displays or hides the breakpoints column, which appears in editor windows to the left of the source code. This command toggles between **Show Breakpoints** and **Hide Breakpoints,** depending on whether the breakpoints column is currently visible on the active editor window.

Set Watchpoint

Clear Watchpoint

Sets or clears a watchpoint for the selected variable or range of memory. This command toggles between **Set Watchpoint** and **Clear Watchpoint**, depending on whether the watchpoint is currently set in the active editor window.

The watchpoint is indicated by an underline. You can configure the color of this underline using the Display Settings preference panel

in the IDE Preferences window. For more information, refer to "Display Settings Panel" on page 271.

Enable Watchpoint

Disable Watchpoint

Enables or disables a watchpoint for the selected variable or range of memory. This command toggles between **Enable Watchpoint** and **Disable Watchpoint**, depending on whether you have a watchpoint currently enabled.

Enabled watchpoints are indicated by an underline for the selected variable or range of memory. Disabled watchpoints do not have the underline. The underline's color can be configured in the Display Settings preference panel of the IDE Preference window. See "Display Settings Panel" on page 271. for more information.

Clear All Watchpoints

Clears all watchpoints in the current program.

Break on C++ Exception

Causes the debugger to break at `__throw()` every time a C++ exception occurs.

Break on Java Exceptions

This menu item causes the Java Exceptions submenu to appear. To learn more about this submenu, refer to "Java Exceptions Submenu" on page 408.

Switch to Monitor

Gives control to an external debugger that you may have installed on your computer.

Mac OS The low-level Macintosh ROM Monitor program and MacsBug are two examples of external debuggers.

For information on this command, see the *Debugger User Guide*.

Data Menu

The Data menu lets you control how data values are displayed in the debugger. All of these commands are available in both the integrated and external CodeWarrior debuggers. This menu is present anytime a window that can employ the Data commands is frontmost on the screen. Otherwise, it is hidden from view.

For more information, see the *Debugger User Guide*.

Show Types

Shows the data types of all local and global variables displayed in the active variable pane or variable window.

New Expression

Creates a new entry in the Expressions window, prompting you to enter a new expression.

Copy to Expression

Copies the variable selected in the active pane to the Expressions window. You can also drag an expression to the Expressions window from source code or from another window or pane.

View As...

Displays a selected variable as a value of a specified data type.

View Variable

Creates a separate window to display a selected variable.

View Array

Creates a separate window to display a selected array.

View Memory

Displays the contents of memory as a hexadecimal/ASCII character dump.

View Memory As...

Displays the memory a selected variable occupies or a selected register points to.

Default

Displays the selected variable in its default format based on the variable type.

Signed Decimal

Displays the selected variable as a signed decimal value.

Unsigned Decimal

Displays the selected variable as an unsigned decimal value.

Hexadecimal

Displays the selected variable as a hexadecimal value.

Character

Displays the selected variable as a character value.

C String

Displays the selected variable as a C character string.

Pascal String

Displays the selected variable as a Pascal character string.

Floating Point

Displays the selected variable as a floating-point value.

Enumeration

Displays the selected variable as an enumeration.

Fixed (Mac OS)

Displays the selected variable as a numerical value of type `Fixed`.

Fract (Mac OS)

Displays the selected variable as a numerical value of type `Fract`.

Window Menu

The Window menu includes commands that tile open editor windows, switch between windows, and open debugger windows. There is also a submenu for customizing the toolbars.

Stack

This command opens all Editor windows to their full screen size and stacks them one on top of another, with their window titles showing. This command is dimmed when the active window is the Project window or Message window.

Tile

This command arranges all Editor windows so that none overlap. This command is dimmed when the active window is the Project window or Message window.

Tile Vertical

This command arranges all the Editor windows in a single row.

This command is disabled when the active window is the Project window or Message window.

Mac OS ToolServer Worksheets also disable this command.

Zoom Window

This menu command expands the active window to the largest possible size. If you choose the menu command again, the window returns to its original size.

Minimize Window (Windows)

Collapse Window (Mac OS)

Minimizes (Windows) or collapses (Mac OS) the currently active window.

Restore Window (Windows)

Expand Window (Mac OS)

Restores (Windows) or expands (Mac OS) a window to its original size.

Save Default Window

This command saves the settings of the active window, so that the next time you open a window of that type, the CodeWarrior IDE opens it with the saved settings. This command works with Browser windows, Message windows, and Editor windows.

To learn more about this command, refer to, "Saving Editor Window Settings" on page 140, or "Saving a Default Browser" on page 233.

Toolbar

This menu item causes the Toolbar submenu to appear. To learn more about this submenu, refer to "Toolbar Submenu" on page 408.

Browser Catalog Window

This command displays the Browser Catalog Window. This menu command is dimmed when the Browser is not activated.

To learn more about this feature, refer to "Catalog Window" on page 209. To learn how to activate the Browser, refer to "Activating the Browser" on page 202.

Class Hierarchy Window

This command displays the Browser's Multi-Class Hierarchy Window. This menu command is dimmed when the Browser is not activated.

To learn more about this feature, refer to "Multi-Class Hierarchy Window" on page 219.

To learn how to activate the Browser, refer to "Activating the Browser" on page 202.

New Class Browser

This command displays the Browser's Multi-Class Browser Window. This menu command is dimmed when the Browser is not activated.

To learn more about this feature, refer to "Multi-Class Browser Window" on page 211.

To learn how to activate the Browser, refer to "Activating the Browser" on page 202.

Build Progress Window

This menu command displays the progress window for builds, as shown in Figure 10.1 on page 321.

Errors & Warnings Window

This command displays the Errors and Warnings window.

To learn more about this window, refer to "Guided Tour of the Message Window" on page 337. Also, refer to "Using Batch Searches" on page 180.

Project Inspector

This menu command allows you to view information about your project and enable debug information generation.

To learn more about this command's window, refer to "Guided Tour of the Project Window" on page 42.

ToolServer Worksheet (Mac OS)

Displays the ToolServer Worksheet window. This window is used in conjunction with ToolServer. This command is disabled provided one of the following conditions have been met:

- ToolServer is not installed on your machine.
- ToolServer is installed on your machine, but has not yet been started. To start ToolServer, select the Start ToolServer command in the Tools menu.

To learn more about ToolServer, refer to "Using MPW ToolServer Overview" on page 513.

Processes Window

Displays the Processes window. For information on this command, see the *Debugger User Guide*.

Expressions Window

Displays the Expressions window. For information on this command, see the *Debugger User Guide*.

Global Variables Window

Displays the Global Variables window. Within this window you can view the global variables for the entire project or those contained in a file. Click on a file name in the Files list to display the file's global variables in the Variables list.

Breakpoints Window

Displays the Breakpoints window. For information on this command, see the *Debugger User Guide*.

Watchpoints Window

Displays the Watchpoints window. For information on this command, see the *Debugger User Guide*.

Registers Windows

Displays the Registers submenu, from which you can choose to view general registers or FPU registers. For information on this command, see the *Debugger User Guide*.

Other Window Menu Items

The other **Window** menu items depend solely on which project, source files, header files, interface files, and other windows you have open.

All of the open windows are shown in this menu and the first nine files (1 through 9) are given key equivalents. The current project is always assigned the number 0 (zero). Press Ctrl/Command and a number to open a specific Editor window. A check mark is placed beside the active window.

Mac OS A file whose modifications have not been saved is underlined.

To make one of your open CodeWarrior files active and bring its window to the front, do one of the following:

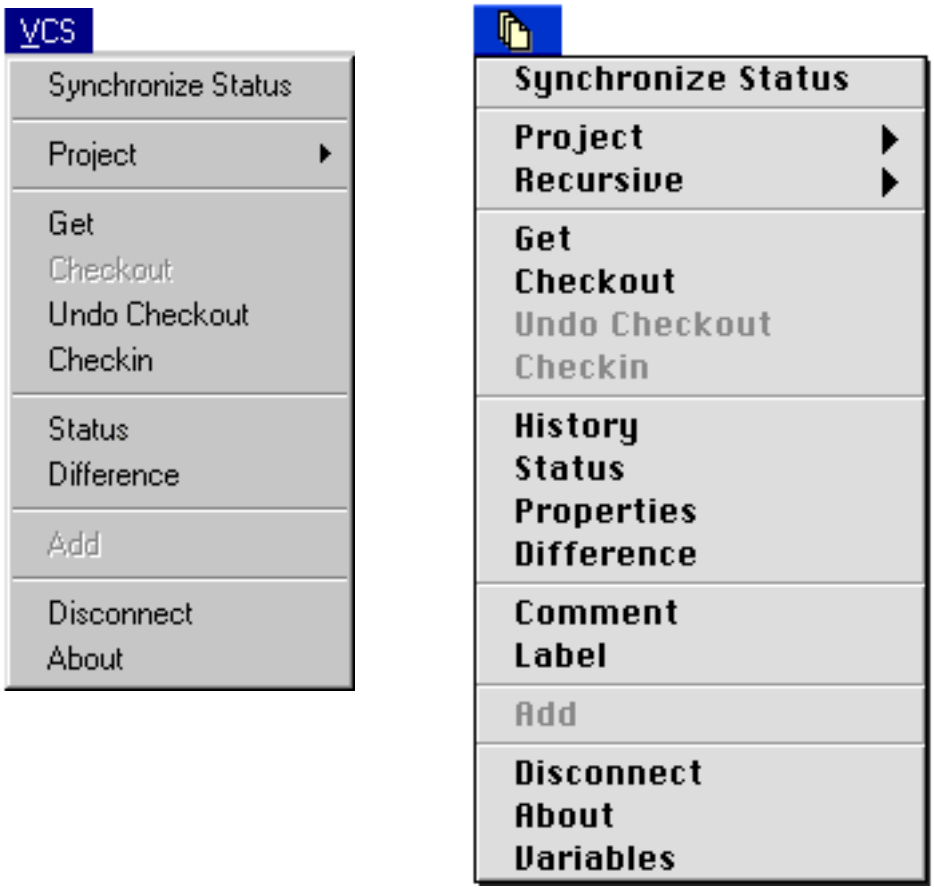
- Click in its window.
- Select it from the Window Menu.
- Use the key equivalent shown in the Window Menu.

Version Control System (VCS) Menu

The Version Control System (VCS) Menu, shown in Figure 12.1, may appear in the menu bar of your CodeWarrior IDE if you have purchased the MW Visual SourceSafe for Macintosh Version Control System (available separately) for use with the CodeWarrior IDE.

To learn how to configure MW Visual SourceSafe to use revision control with your projects, refer to the documentation for your MW Visual SourceSafe product.

Figure 12.1 VCS menus for Windows and Mac OS



Help Menu

Online help is available from the Help menu. When you are working in the CodeWarrior IDE, select one of the items to get interactive, online help.

Windows Help Menu

The Help menu contains a list of help files that can assist you in understanding and using CodeWarrior. Choosing an item displays the relevant help files. Each item covers a specific topic.

Contents (Windows)

This menu command displays the CodeWarrior help files.

Keys (Windows)

This menu command displays the topic index for all CodeWarrior online help files.

CodeWarrior IDE (Windows)

This menu command displays the topic index for the CodeWarrior IDE online help file.

Debugger (Windows)

This menu command displays the topic index for the integrated debugger online help file.

How to Use Help (Windows)

This menu command displays the online files that describe how to use the help facilities.

About Metrowerks (Windows)

Displays the Metrowerks About Box.

Mac OS Help Menu

The **Help** menu under Mac OS 8 or newer contains a list of Apple Guides that can assist you in understanding and using the CodeWarrior IDE. Choosing an item displays an Apple Guide topic window. Each Apple Guide covers a specific topic.

Java Exceptions Submenu

The Debug Menu has another submenu under it for the **Break on Java Exceptions** command. The Java Exceptions submenu contains options that tell the integrated debugger what to do when Java exceptions occur. This command is available when you are working with Java source code, and is dimmed for all other source code.

All Exceptions

This menu command causes the debugger to break every time an exception occurs. This includes exceptions thrown by the virtual machine, your own classes, the debugger, classes in `classes.zip`, and similar exceptions. Java throws many exceptions in the normal course of execution, so catching all exceptions causes the debugger to break often.

Exceptions in Targeted Classes

This menu command causes the debugger to break only on exceptions that are thrown by your own classes in the project. Select this menu command if you are interested in breaking on the exceptions thrown by your classes, rather than the exceptions that Java throws in the normal course of execution.

No Exceptions

When you select this menu command, the debugger will not break when exceptions occur.

Toolbar Submenu

The Window Menu has another submenu under it for the **Toolbar** command. The Toolbar submenu contains all the commands used to customize the toolbars that appear in CodeWarrior IDE windows.

To learn more about how to customize the toolbars, read the information in "Customizing Toolbars" on page 280.

Toolbar Elements Window

The menu command shows the Toolbar Elements window. From this window you can customize the toolbars by adding icon shortcuts to better suit the way you work.

Show Window Toolbar and Hide Window Toolbar

These menu commands cause the toolbar in the active window to disappear or reappear. The actual command shown in the menu will toggle between **Show Window Toolbar** and **Hide Window Toolbar**, depending on whether the active window's toolbar is visible or not.

Reset Window Toolbar

This menu command causes the toolbar in the active window to reset to a default state. You should use this menu command if you want to return the Editor window toolbar to its original default settings.

Clear Window Toolbar

This menu command causes the toolbar in the active Editor, Project, or Browser window to have all icons removed from it. Once all the icons have been removed, you can add icons using the Toolbar Elements window.

Use the **Reset Window Toolbar** command to cause all the default icons to come back.

Show Floating Toolbar and Hide Floating Toolbar

These menu commands cause the Floating Toolbar to appear or disappear. The actual command shown in the menu will toggle between **Show Floating Toolbar** and **Hide Floating Toolbar**, depending on whether the Floating Toolbar is already visible or not.

Anchor Floating Toolbar and Unanchor Floating Toolbar (Mac OS)

These menu commands cause the Floating Toolbar to anchor and unanchor itself to the left edge of the screen, immediately below the menu bar. The actual command shown in the menu will toggle between **Anchor Floating Toolbar** and **Unanchor Floating Toolbar**, depending on whether the Floating Toolbar is currently anchored or not.

Reset Floating Toolbar

This menu command causes the Floating Toolbar to return to its default state. You should use this menu command if you want to return the Floating Toolbar to the original default settings.

Clear Floating Toolbar

This menu command causes the Floating Toolbar to have all icons removed from it. Once all the icons have been removed, you can add icons using the Toolbar Elements window.

Apple Menu (Mac OS)

The Apple menu contains one IDE-related item.

About Metrowerks…

Choose this item to see the way-cool About Box.

Tools Menu (Mac OS)

The Tools menu contains commands used to do things like start and stop ToolServer. It also contains commands that execute Macintosh Programmer's Workbench (MPW) tools and scripts. Additional commands reference the MPW 411 database.

To learn more about how to use the CodeWarrior command-line tools, refer to the *CodeWarrior Command Line Tools* manual for more information.

To learn more about ToolServer, refer to "Using MPW ToolServer Overview" on page 513.

Start ToolServer

Initiates ToolServer handshaking. If you do not have ToolServer installed on your computer, this command is replaced by **Can't Find ToolServer**.

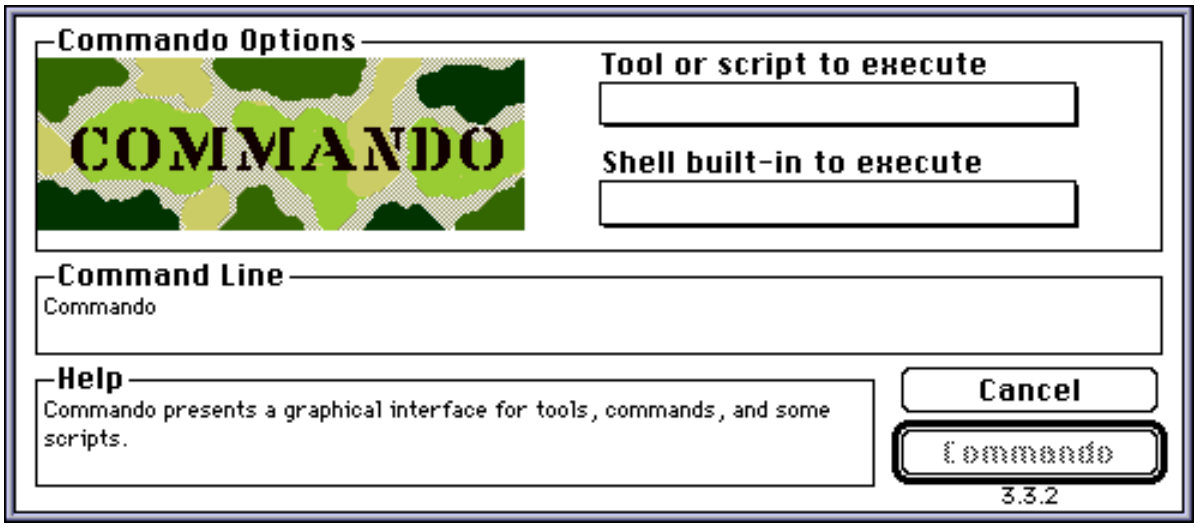
Stop ToolServer

When ToolServer is launched, the **Stop ToolServer** command replaces the **Start ToolServer** command. Stop ToolServer quits ToolServer and removes the extra ToolServer menu from the CodeWarrior IDE's menu bar.

Commando

This command brings up the Commando dialog box, as shown in Figure 12.2. To learn more about Commando, refer to "Using MPW ToolServer Overview" on page 513.

Figure 12.2 Commando Dialog Box



Execute as ToolServer Script

Executes the ToolServer script in the active editor window. For more information, see "Using the ToolServer Tools Menu" on page 521.

Lookup Symbol

Searches for information about the selected symbol in the MPW 411 database. For more information, see "Looking Up MPW 411 Documentation" on page 528.

Insert Template

Inserts the parameter list template for a function after the selected text. For more information, see "Looking Up MPW 411 Documentation" on page 528.

ToolServer Tools

You can use this submenu to run an MPW tool or script that is installed in the Tools folder of your ToolServer application. To learn more about ToolServer, refer to "Using MPW ToolServer Overview" on page 513.

Scripts Menu (Mac OS)

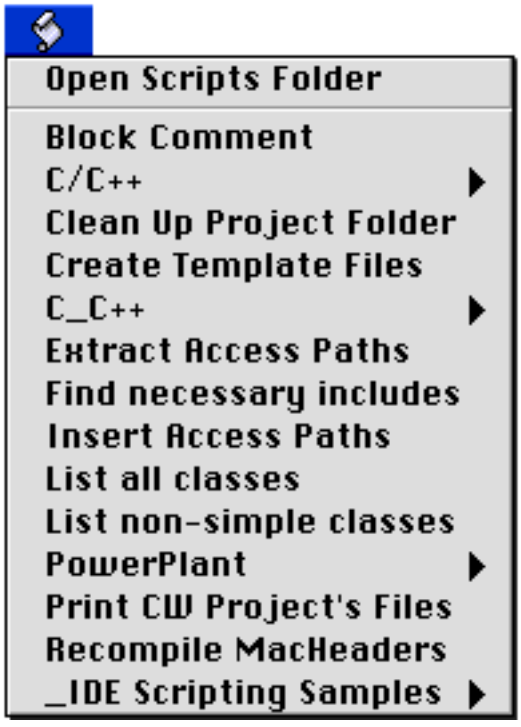
The Script menu, shown in Figure 12.3, contains a list of the AppleScripts in the `(Scripts)` folder.

This menu will only appear if the **Use Script Menu** setting is turned on in the IDE Extras preference panel. Also, this menu will only be shown if the `(Scripts)` folder is present in your Metrowerks CodeWarrior folder. To learn more about how to configure the appropriate preferences, refer to "IDE Extras Panel" on page 242.

To learn more about AppleScripts and scripting the CodeWarrior IDE, refer to "CodeWarrior Apple Events Overview" on page 433.

To learn about what the individual scripts do, open them with a text editor or an AppleScript editor application.

Figure 12.3 The Script menu



Note that the menu is hierarchical, based on the directory structure of the `(Scripts)` folder. You may place scripts in the `(Scripts)` folder inside other directories to create a hierarchical menu.

Open Scripts Folder

Brings the Finder to the front and opens the `(Scripts)` folder. This command is always available if the Script menu is active.

Other Script menu items

The other items in this menu are the names of the AppleScripts in the `(Scrips)` folder. If you do not have any AppleScripts, no items are listed.

Editor Extensions Menu (Mac OS)

The Editor Extensions menu (Figure 12.4) is similar to the Script Menu. If the Use BBEdit™ Extensions preference is enabled, a list of BBEdit™ extensions are added to the menu bar. You must have an alias or folder called `(Editor Extensions)`, including the parentheses, in the same folder as the CodeWarrior IDE application. Place the editor extensions that you wish to use inside the `(Editor Extensions)` folder. You can also place aliases of the editor extensions inside the `(Editor Extensions)` folder.

To learn more about how to configure the appropriate preferences so this menu becomes visible, refer to "IDE Extras Panel" on page 242.

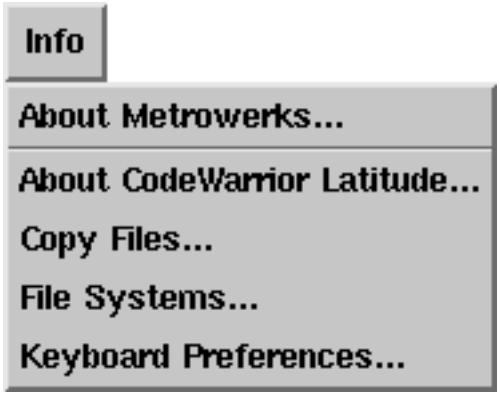
For more information about BBEdit, refer to the documentation that came with the product.

Figure 12.4 Editor Extensions menu

Info Menu (Solaris)

The Info menu (Figure 12.5) is contains the following menu items:

Figure 12.5 Solaris Info men



About Metrowerks...

Choose this item to see the Metrowerks CodeWarrior about box.

About CodeWarrior Latitude...

Choose this item to see the CodeWarrior Latitude about box.

Copy Files...

This item launches a desk accessory which enables you to copy files and folders. From within the CodeWarrior IDE, you can copy files and folders between file systems or devices accessible from your Solaris environment. See "Copy Files Accessory" on page 531. for a detailed description of this accessory.

File Systems...

This facility allows you to mount volumes from within the CodeWarrior IDE. For more information on using this facility, refer to "File Systems Facility" on page 535.

Keyboard Preferences...

Use this command to display the Keyboard Preferences dialog box where you can change the default modifier key mappings. To learn more about this settings panel, refer to "Keyboard Preferences Dialog Box" on page 539.

A

Default CodeWarrior Key Bindings

This chapter describes the default key bindings assigned to commands in the CodeWarrior IDE.

Some commands do not have any key bindings assigned to them by CodeWarrior, so their respective cells appear blank. You can assign key bindings to any blank command. For more information on key bindings, see "Key Bindings Panel" on page 247.

The key bindings sections include:

- File Menu
- Edit Menu
- Search Menu
- Project Menu
- Debug Menu
- Data Menu
- Window Menu
- Misc.
- Editor Commands
- Prefix Keys

Mac OS and Solaris Modifier Key Legend

Modifier keys are used in combination with other keys to generate keyboard shortcuts to certain IDE functions. For instance, to execute the Ctrl/Command - N keyboard shortcut, you first depress the Ctrl or Command modifier key, and while holding it down, press the N key on your keyboard. For Mac OS- and Solaris-hosted versions of

the CodeWarrior IDE, symbols are used to represent modifier keys in menus as well as the Key Bindings preferences panel. The legend in Table A.1 describes how these symbols correspond to the names associated with them on both Solaris and Mac OS.

Table A.1 Mac OS and Solaris modifier key legend

Symbol	Mac OS	Solaris
⌘	Command key	Meta key
⌥	Option key	Alt key
⇧	Shift key	Shift key
⌃	Control key	Control key

NOTE: Solaris modifier key mappings can be changed via the Keyboard Preferences dialog box. See "Keyboard Preferences Dialog Box" on page 539 for more information on changing the default modifier key mappings.

File Menu

Table A.2 contains the default key bindings for manipulating projects and files from within the CodeWarrior IDE.

Table A.2 File key bindings

Command	Windows	Mac OS	Solaris
New	Ctrl-N	Command-N	Command-N
New Project	Ctrl-Shift-N	Shift-Command-N	Shift-Command-N

Command	Windows	Mac OS	Solaris
New Empty Project	Ctrl-Alt-Shift-N	Option-Shift-Command-N	Option-Shift-Command-N
Open	Ctrl-O	Command-O	Command-O
Find and Open 'selection'	Ctrl-D	Command-D	Command-D
Find and Open File	Ctrl-Shift-D	Option-Command-D	Option-Command-D
Close	Ctrl-W	Command-W	Command-W
Close All	Ctrl-Shift-W	Option-Command-W	Option-Command-W
Switch to MW Debugger		Command-J	Command-J
Save	Ctrl-S	Command-S	Command-S
Save All	Ctrl-Shift-S	Option-Command-S	Option-Command-S
Save As			
Save A Copy As			
Revert			
Page Setup			
Print	Ctrl-P	Command-P	Command-P
Quit		Command-Q	Command-Q

Edit Menu

Table A.3 contains the default key bindings for the commands in the Edit menu of the CodeWarrior IDE.

Table A.3 Edit key bindings

Command	Windows	Mac OS	Solaris
Undo	Ctrl-Z	Command-Z	Command-Z
Redo	Ctrl-Shift-Z	Shift-Command-Z	Shift-Command-Z
Cut	Ctrl-X	Command-X	Command-X
Copy	Ctrl-C	Command-C	Command-C
Paste	Ctrl-V	Command-V	Command-V
Clear		Clear	
Select All	Ctrl-A	Command-A	Command-A
Balance	Ctrl-B	Command-B	Command-B
Shift Left	Ctrl-[	Command-[	Command-[
Shift Right	Ctrl-]	Command-]	Command-]
Insert Reference Template		Command-Y	Command-Y
Preferences			
Target Settings	Alt-F7		
VCS Settings			

Search Menu

Table A.4 contains the default key bindings for the commands in the Search menu of the CodeWarrior IDE.

Table A.4 Search key bindings

Command	Windows	Mac OS	Solaris
Find	Ctrl-F	Command-F	Command-F
Find Next	F3	Command-G	Command-G
Find Previous	Shift-F3	Shift-Command-G	Shift-Command-G
Find in Next File	Ctrl-T	Command-T	Command-T
Find in Previous File	Ctrl-Shift-T	Shift-Command-T	Shift-Command-T
Enter Find String	Ctrl-E	Command-E	Command-E
Enter Replace String	Ctrl-Shift-E	Shift-Command-E	Shift-Command-E
Find Selection	Ctrl-F3	Command-H	Command-H
Find Previous Selection	Ctrl-Shift-F3	Shift-Command-H	Shift-Command-H
Replace	Ctrl-=	Command-=	Command-=
Replace & Find Next	Ctrl-L	Command-L	Command-L
Replace & Find Previous	Ctrl-Shift-L	Shift-Command-L	Shift-Command-L
Replace All			
Find Definition	Ctrl-’	Command-’	Command-’
Find Definition & Reference		Option-Command-’	Option-Command-’

Command	Windows	Mac OS	Solaris
Find Reference		Shift-Command-’	Shift-Command-’
Go Back	Ctrl-Shift-B	Shift-Command-B	Shift-Command-B
Go Forward	Ctrl-Shift-F	Shift-Command-F	Shift-Command-F
Goto Line	Ctrl-G	Command-,	Command-,
Compare Files			
Apply Difference			
Unapply Difference			

Project Menu

Table A.5 contains the default key bindings for managing projects, controlling compilations, and much more.

Table A.5 Project key bindings

Command	Windows	Mac OS	Solaris
Add Window			
Add Files			
Create Group/ Segment/Target			
Remove Selected Items	Ctrl-Del	Command-Del	Command-Del
Check Syntax	Ctrl-;	Command-;	Command-;
Preprocess			
Precompile			

Command	Windows	Mac OS	Solaris
Compile	Ctrl-F7	Command-K	Command-K
Disassemble			
Bring Up To Date		Command-U	Command-U
Make	F7	Command-M	Command-M
Remove Object Code	Ctrl - –	Command - –	Command - –
Remove Object Code & Compact	Ctrl - Shift - –	Option - Command - –	Option - Command - –
Re-search For Files			
Reset Project Entry Paths			
Synchronize Modification Dates			
Enable Debugging			
Run/Debug	F5	Command-R	Command-R
Debug/Run	Ctrl-F5	Option-Command-R	Option-Command-R

Debug Menu

Table A.6 contains the default key bindings for handling commands on the Debug menu in the CodeWarrior IDE.

Table A.6 Debug menu key bindings

Command	Windows	Mac OS	Solaris
Kill	Shift-F5	Control-K	Control-K
Reset			

Command	Windows	Mac OS	Solaris
Hard Reset			
Step Over	F10	Control-S	Control-S
Step Into	F11	Control-T	Control-T
Step Out	Shift-F11	Control-U	Control-U
Stop		Control-P	Control-P
Set/Clear Breakpoint			
Enable/Disable Breakpoint			
Clear All Breakpoints			
Show/Hide Breakpoints			
Set/Clear Watchpoint			
Enable/Disable Watchpoint			
Clear All Watchpoints			
Switch to Monitor			

Data Menu

Table A.7 contains the default key bindings for handling debugger variable displays in the IDE.

Table A.7 Data menu key bindings

Command	Windows	Mac OS	Solaris
Show Types			
New Expression	Alt-N	Control-N	Control-N
Copy To Expression			
View As	Ctrl-Y	Control-Y	Control-Y
View Variable			
View Array		Control-A	Control-A
View Memory	Alt-Shift-6	Control-M	Control-M
View Memory As			
View As Default			
View As Signed Decimal	Alt-Shift-D	Control-Shift-D	Control-Shift-D
View As Unsigned Decimal	Alt-Shift-U	Control-Shift-U	Control-Shift-U
View As Hexadecimal	Alt-Shift-H	Control-Shift-H	Control-Shift-H
View As Character	Alt-Shift-C	Control-Shift-C	Control-Shift-C
View As C String	Alt-Shift-S	Control-Shift-S	Control-Shift-S
View As Pascal String	Alt-Shift-P	Control-Shift-P	Control-Shift-P
View As Floating Point	Alt-Shift-F	Control-Shift-F	Control-Shift-F

Command	Windows	Mac OS	Solaris
View As Enumeration	Alt-Shift-E	Control-Shift-E	Control-Shift-E
Fixed		Control-Shift-I	Control-Shift-I
Fract		Control-Shift-R	Control-Shift-R

Window Menu

Table A.8 contains the default key bindings for handling many common windows in the CodeWarrior IDE.

Table A.8 Window menu key bindings

Command	Windows	Mac OS	Solaris
Stack			
Tile			
Tile Vertical			
Zoom Window	Ctrl-/	Command-/	Command-/
Collapse/Expand Window			
Save Default Window			
Browser Catalog Window			
Class Hierarchy Window			
New Class Browser	Alt-F12		
Build Progress Window			

Command	Windows	Mac OS	Solaris
Errors & Warnings Window	Ctrl-I	Command-I	Command-I
Project Inspector	Alt-Enter		
ToolServer Worksheet			
Processes Window			
Expressions Window	Alt-Shift-3		
Global Variables Window			
Breakpoints Window	Alt-F9		
Watchpoints Window			
Select Default Project	Ctrl-0	Command-0	Command-0
Select Document 1	Ctrl-1	Command-1	Command-1
Select Document 2	Ctrl-2	Command-2	Command-2
Select Document 3	Ctrl-3	Command-3	Command-3
Select Document 4	Ctrl-4	Command-4	Command-4
Select Document 5	Ctrl-5	Command-5	Command-5
Select Document 6	Ctrl-6	Command-6	Command-6
Select Document 7	Ctrl-7	Command-7	Command-7
Select Document 8	Ctrl-8	Command-8	Command-8
Select Document 9	Ctrl-9	Command-9	Command-9

Misc.

Table A.9 contains the default key bindings for handling miscellaneous tasks in the CodeWarrior IDE.

Table A.9 Misc. key bindings

Command	Windows	Mac OS	Solaris
Go to Header / Source File	Ctrl-`	Command-Tab	Command-Tab
Go to Previous Error Message	F4	Option-Command-Up Arrow	Option-Command-Up Arrow
Go to Next Error Message	Shift-F4	Option-Command-Down Arrow	Option-Command-Down Arrow
Run Script			
Stop Script			

Editor Commands

Table A.10 contains the default key bindings for handling Editor windows in the CodeWarrior IDE.

Table A.10 Editor window key bindings

Command	Windows	Mac OS	Solaris
Move Character Left	Left Arrow	Left Arrow	Left Arrow
Move Character Right	Right Arrow	Right Arrow	Right Arrow
Move Word Left	Ctrl-Left Arrow	Option-Left Arrow	Option-Left Arrow

Command	Windows	Mac OS	Solaris
Move Word Right	Ctrl-Right Arrow	Option-Right Arrow	Option-Right Arrow
Move Sub-word Left	Alt-Left Arrow	Control-Left Arrow	Control-Left Arrow
Move Sub-word Right	Alt-Right Arrow	Control-Right Arrow	Control-Right Arrow
Move to Start of Line	Home	Command-Left Arrow	Command-Left Arrow
Move to End of Line	End	Command-Right Arrow	Command-Right Arrow
Move Line Up	Up Arrow	Up Arrow	Up Arrow
Move Line Down	Down Arrow	Down Arrow	Down Arrow
Move to Top of Page	Page Up	Option-Up Arrow	Option-Up Arrow
Move to Bottom of Page	Page Down	Option-Down Arrow	Option-Down Arrow
Move to Top of File	Ctrl-Home	Command-Up Arrow	Command-Up Arrow
Move to Bottom of File	Ctrl-End	Command-Down Arrow	Command-Down Arrow
Delete Character Left	Backspace	Delete	Delete
Delete Character Right	Del	del	del
Delete to End of File		Command-Delete	Command-Delete
Character Select Left	Shift-Left Arrow	Shift-Left Arrow	Shift-Left Arrow

Command	Windows	Mac OS	Solaris
Character Select Right	Shift-Right Arrow	Shift-Right Arrow	Shift-Right Arrow
Select Word Left	Ctrl-Shift-Left Arrow	Option-Shift-Left Arrow	Option-Shift-Left Arrow
Select Word Right	Ctrl-Shift-Right Arrow	Option-Shift-Right Arrow	Option-Shift-Right Arrow
Select Sub-word Left	Alt-Shift-Left Arrow	Control-Shift-Left Arrow	Control-Shift-Left Arrow
Select Sub-word Right	Alt-Shift-Right Arrow	Control-Shift-Right Arrow	Control-Shift-Right Arrow
Select Line Up	Shift-Up Arrow	Shift-Up Arrow	Shift-Up Arrow
Select Line Down	Shift-Down Arrow	Shift-Down Arrow	Shift-Down Arrow
Select to Start of Line	Shift-Home	Shift-Command-Left Arrow	Shift-Command-Left Arrow
Select to End of Line	Shift-End	Shift-Command-Right Arrow	Shift-Command-Right Arrow
Select to Start of Page	Shift-Page Up	Option-Shift-Up Arrow	Option-Shift-Up Arrow
Select to End of Page	Shift-Page Down	Option-Shift-Down Arrow	Option-Shift-Down Arrow
Select to Start of File	Ctrl-Shift-Home	Shift-Command-Up Arrow	Shift-Command-Up Arrow
Select to End of File	Ctrl-Shift-End	Shift-Command-Down Arrow	Shift-Command-Down Arrow
Scroll Line Up	Ctrl-Up Arrow	Control-Up Arrow	Control-Up Arrow
Scroll Line Down	Ctrl-Down Arrow	Control-Down Arrow	Control-Down Arrow
Scroll Page Up		Page Up	Page Up
Scroll Page Down		Page Down	Page Down

Command	Windows	Mac OS	Solaris
Scroll to Top of File		Home	Home
Scroll to End of File		End	End
Scroll to Selection			
Find Symbols with Prefix	Ctrl-\	Control-\	Control-\
Find Symbols with Substring	Ctrl-Shift-\	Control-Shift-\	Control-Shift-\
Get Next Symbol	Ctrl-.	Control-.	Control-.
Get Previous Symbol	Ctrl-,	Control-,	Control-,

Prefix Keys

Table A.11 contains the default key bindings for handling prefixes.

Table A.11 Prefix key bindings

Command	Windows	Mac OS	Solaris
Quote Key			
Prefix Key 1			
Prefix Key 2			
Prefix Key 3			

B

Mac OS CodeWarrior Scripting

This chapter introduces and discusses the Apple Event and AppleScript support provided by the CodeWarrior IDE.

CodeWarrior Apple Events Overview

This chapter discusses the AppleScript and Apple Event commands and classes supported in CodeWarrior. You should read this chapter if you would like to enhance and extend the capabilities of the CodeWarrior IDE.

The CodeWarrior IDE supports Apple Events. By scripting these Apple Events using AppleScript or another scripting editor, such as Frontier, it is possible to execute many CodeWarrior IDE commands without using the IDE directly. Scripting the CodeWarrior IDE is a way to automate repetitive tasks that do not need user interaction.There are many exciting things that you can do with AppleScript to harness the power of the IDE, such as automate builds, generate files automatically, and configure settings.

If you are primarily interested in writing scripts that manipulate and automate the IDE, then you are probably most interested in using AppleScript to put together an ensemble of Apple Events. If you would like to write program code to drive the CodeWarrior IDE from within your own computer program or tools, then you are probably most interested in the lower-levels of Apple Events, and not in AppleScript. This chapter is oriented toward working with AppleScript, but there is a discussion of low-level Apple Event cod-

ing in "Coding with CodeWarrior IDE and Apple Events" on page 511.

TIP: Look at the AppleScripts in the `(Scripts)` folder of the `Metrowerks CodeWarrior` folder for lots of cool AppleScripts. Reviewing these scripts will save you time when learning to write your own.

This chapter is not a tutorial. If you want to learn how to edit, save, and run AppleScripts, you will not the information here. Instead, refer to other tools and sources of information listed in "AppleScript Tools and Reference Material" on page 434 for more information.

The topics in this chapter are:

- AppleScript Tools and Reference Material
- Writing Your First CodeWarrior IDE AppleScript
- CodeWarrior IDE AppleScript Events
- CodeWarrior IDE AppleScript Classes
- Coding with CodeWarrior IDE and Apple Events

TIP: You can run AppleScripts from within the CodeWarrior IDE. To learn about how to do this, read the section of this manual entitled "Scripts Menu (Mac OS)" on page 413.

AppleScript Tools and Reference Material

You can find the tools provided by Apple Computer for editing and running AppleScripts on the CodeWarrior Reference CD in the `MacOS System Extensions` folder. Use the installer provided there to install the tools on your system.

Other Editing and debugging tools are available from third-party vendors. These products are worth evaluating if you are going to do

much AppleScripting. The following is neither an exhaustive list nor an endorsement of these products.

- Script Debugger (Late Night Software)
- Scripter 2.0 (Main Event Software)

If you are a subscriber to the Apple Developer CD program or Apple Developer Mailing, you will find good information on Apple Events and AppleScripting on the CDs.

For more information on using and writing AppleScripts, you may want to consult other publications, such as:

- *AppleScript Language Guide: English Dialect* (Addison-Wesley)
- *Danny Goodman's AppleScript Handbook* (Random House)
- *The Tao of AppleScript* (Hayden Books)
- *Applied Mac Scripting* (M & T Books)

For information on more advanced topics such as writing your own Scripting Additions, or how to use the standard Scripting Additions, refer to *AppleScript Scripting Additions Guide* (Apple Computer).

On the internet, Apple Computer maintains a web site for AppleScript issues, as well as email lists of AppleScript topics. Point your web browser at `http://AppleScript.apple.com` to learn more.

Finally, for documentation on using low-level AppleEvents in program code, refer to *Inside Macintosh: Interapplication Communication* (Addison-Wesley).

Writing Your First CodeWarrior IDE AppleScript

To get started with AppleScript and the CodeWarrior IDE, let's take a look at a simple script that opens the IDE, brings it to the foreground on the Mac, opens a project, removes the binaries, and starts a build of the project. This script, shown in Listing B.1, is something that could be double-clicked to automatically do all these operations unattended.

Listing B.1 My First CodeWarrior AppleScript

```
tell application "CodeWarrior IDE 3.2" (* go! *)
  activate (* bring CW to the front *)
  open file "SD:MyProj:MyProject.68K.µ"
  Remove Binaries
  Make Project
end tell
```

You can imagine how convenient it will be to automate many tasks with AppleScript from this short example. Try entering this script in your Editor, such as Apple's Script Editor that comes with the AppleScript 1.1 software on the CodeWarrior Reference CD, and get it to run.

After getting this short example to run, you will probably be motivated to try some more extensive examples.

CodeWarrior IDE AppleScript Events

In general, Apple Events are grouped in categories or "suites" of events that provide some common theme for the events. There is a "Required" suite of events that includes open, print, quit and run. All scriptable applications should support the required suite. There are other suites of events defined in the *Apple Event Registry* document. In addition, there are other suites of events that are application-specific.

For many of the things that you probably want to do with the CodeWarrior IDE, it really isn't a concern which suite an event is from most of the time. However, you can view the "dictionary" of Apple Events that an application supports using the Open Dictionary command of your Script Editor. See the documentation that came with your editor for information about viewing the dictionary.

In this section, we discuss how to handle errors in AppleScript, and several categories of events that you can use to control the CodeWarrior IDE.

- Processing Errors

- Required Events
- File Handling Events
- Building Events
- Status/Query Events
- Navigation Events

Parameters

Some Apple Events listed in this section require a parameter called *filename-list*. The *filename-list* represents a single filename or a list of filenames and/or aliases. A single filename is a quoted character string. A list of filenames is enclosed in braces, {}, with the filenames separated by commas.

Listing B.2 Example values for *filename*

```
"myprogram.c"
{"startup.p", "printout.p", "drawbox.p"}
{"codechecker.cpp"}
{"HD:CodeWarrior ƒ:My Projects:hello.c"}
file "myprogram.c"
alias "myprogram.c"
```

Processing Errors

When an AppleEvent is sent to CodeWarrior, errors may be returned to the script. Errors are not always evil occurrences, as sometimes you will want to trap errors to make your script do other things in response to the current conditions. For example, you can trap a file-not-found error and direct the CodeWarrior IDE to perform an alternate action as a result. Errors can be generated from the operating system, or from the application you're trying to script. Errors for the operating system are documented in Appendix C of the *AppleScript Language Guide*. Errors generated by the CodeWarrior IDE are documented here.

Errors are usually returned through the normal Error-return channel. However, for events that process a list of files, the errors are re-

turned in `the result` (a built-in AppleScript variable). The list, with each member corresponding to an input file, is returned as the event's result with each list member.

In addition to operating system errors, such as out of memory errors, the error codes listed in Table B.1 may also be returned.

Table B.1 CodeWarrior keyAEResult result codes (typeShortInteger)

Name	Value
`noErr`	`0`
`errShell_ActionFailed`	`1`
`errShell_FileNotFound`	`2`
`errShell_DuplicateFile`	`3`
`errShell_CompileError`	`4`
`errShell_MakeFailed` (compile or link error)	`5`
`errShell_NoOpenProject`	`6`
`errShell_WindowNotOpen`	`7`
`errShell_SegmentNotFound`	`8`

The result parameter, `keyAEResult`, is not set if there is an error while interpreting the AppleEvent (running out of memory, supplying a bad parameter type, and so on). In such cases, an error code is returned in the standard `keyErrorNumber` parameter.

Listing B.3 gives an example of an AppleScript that handles an error.

Listing B.3 Error handling in AppleScript

```
try
  tell application "CodeWarrior IDE 3.2"
    set doclist to (Get Open Documents)
  end tell
```

```
on error number errnum
  display dialog "Bummer!" & errnum
end try
```

To see more examples of error handling in scripts, review some of the scripts in the `(Scripts)` folder in the same folder as your CodeWarrior IDE application.

Required Events

There are four events that are required for every application that claims to be AppleScriptable. This section discusses these four events and their syntax.

The events covered in this section are:

- Open
- Print
- Quit
- Run

Open

Purpose This event tells the CodeWarrior IDE to open the specified files.

```
Open filename-list [ converting ] expression
```

If `converting` is specified, any project files that were created with previous versions of the CodeWarrior IDE will be updated.

Listing B.4 Example for Open

```
Open "HD:MyProject.µ" converting yes
Open "HD:MyProject.µ"
```

Print

Purpose This event tells the CodeWarrior IDE to print the specified files.

`Print` *filename-list*

Quit

Purpose This event tells the CodeWarrior IDE to quit.

Run

Purpose This event is sent to an application when it is double-clicked. Upon receiving the event, the application should launch itself.

File Handling Events

You will want to use the File Handling Apple Events of the CodeWarrior IDE to do things like add and remove files in a project, close a window, create and close a project, and save copies of files.

Here are the events covered in this section:

- Add Files
- Add New Files
- Close Project
- Close Window
- Create Project
- Remove Files
- Remove Target Files
- Save Error Window As
- Select
- Set Modification Date

Add Files

Purpose Adds the specified files to the current project.

`Add Files` *filename-list* `[ to segment` *number* `]`

Description This event is equivalent to the **Add Files** command in the Project Menu. The *filename-list* parameter describes a single filename or list of filenames to add the current project. The optional `to segment` parameter specifies the segment in the project in which to add the files. Replace *number* with the segment number to place the files in. The default is to create a new segment.

Returns A list of errors. The result code for each file added to the project can either return the value of an OSErr (Operating System Error) or one of the following values:

- `noerr`
- `errShell_FileNotFound`
- `errShell_DuplicateFile`
- `errShell_NoOpenProject`

Listing B.5 Examples for Add Files

```
Add Files "MyFile.c"
Add Files "MyFile.c" to segment 2
Add Files {"MyFile.c", "MyFile2.c"}
Add Files {"MyFile.c", "MyFile2.c"} to segment 3
```

Add New Files

Purpose Adds the specified files to the current project or target.

```
Add New Files filename-list  [ to segment number ]
```

Description This event is equivalent to the **Add Files** command in the Project Menu. The *filename-list* parameter describes a single filename or list of filenames to add the current project. The optional `to segment` parameter specifies the segment in the project in which to add the files. Replace *number* with the segment number to place the files in. The default is to create a new segment.

Returns A list of errors. The result code for each file added to the project can either return the value of an OSErr (Operating System Error) or one of the following values:

- `noerr`
- `errShell_FileNotFound`
- `errShell_DuplicateFile`
- `errShell_NoOpenProject`

Listing B.6 Examples for Add New Files

```
Add New Files "MyFile.c"
Add New Files "MyFile.c" to segment 2
Add New Files {"MyFile.c", "MyFile2.c"}
Add New Files {"MyFile.c", "MyFile2.c"} to segment 3
```

Close Project

Purpose Closes the current project.

```
Close Project
```

Returns None.

Close Window

Purpose Closes editor windows.

```
Close Window filename [ saving status ]
```

Description This event is equivalent to the **Close** command in the File Menu. If *filename* is a string, Close Window first tries to find the first matching window name, searching front to back. If no window name matches *filename*, then Close Window causes a search for a matching filename.

To specify read/only windows, add " [r/o 1]" to the end of *filename*.

```
Close Window "hello.c [r/o 1]"
```

Table B.2 lists file saving options used with the close window AppleScript command.

Table B.2 Saving options ('savo')

Name	Property Code
`yes` (Save changes)	`'yes '`
`no` (Do not save changes)	`'no  '`
`ask` (Ask the user whether to save)	`'ask '`

The optional `saving` parameter determines if the windows contents are saved before closing the window.

- If *status* is `yes`, save the window's contents.
- If *status* is `no`, discard changes made to the file.
- If *status* is `ask`, prompt the user whether or not to save the file.

Returns None.

Listing B.7 Example for Close Window

```
Close Window "untitled"
    -- Closes first "untitled" window.
Close Window "hello.c" saving yes
Close Window "main.c [r/o 1]"
    -- Closes read/only window.
```

Create Project

Purpose Creates a new project file.

```
Create Project filename [ from stationery ]alias
```

Description Performs the **New Project** command in the File Menu. Replace *filename* with a single filename for the new project. If `from stationery` is specified, project stationery specified by *alias* is used to create the new project.

Returns The result code can have the following values:

- `noErr`
- `errShell_ActionFailed`

Listing B.8 Examples for Create Project

```
Create Project "HardDisk:Projects:MyProject.µ"
Create Project "Foobe" from ¬
  stationery "HD:Dev:Metrowerks" & ¬
  "CodeWarrior:(Project Stationery):MacOS:C/C++:" & ¬
  "Basic Toolbox 68k:Basic Toolbox 68k.µ"
Create Project "::sample project:sample.µ"
```

Remove Files

Purpose Removes the specified file(s) from the current project.

`Remove Files` *filename*

Description Performs the equivalent of the **Remove Selected Items** command in the Project Menu. Replace *filename* with a single file or a list of files to remove from the current project.

Returns A list of errors. The result code can have one of the following values:

- noErr
- errShell_FileNotFound
- errShell_NoOpenProject

Listing B.9 Examples for Remove Files

```
Remove Files "MyFile.c"
Remove Files { "MyFile.c", "YourFile.c" }
```

Remove Target Files

Purpose Removes files from the current target or project.

```
Remove Target Files filename-list
```

Description Performs the equivalent of the **Remove Selected Items** command in the Project Menu.

Returns None.

Save Error Window As

Purpose Saves the contents of the message window as a text file.

```
Save Error Window as filename
```

Description Performs the equivalent of the **Save A Copy As** command in the File Menu when the Message window is active. The contents of the Message window are saved with the name *filename*.

Returns None.

Listing B.10 Example for Save Error Window As

```
Precompile "main.pch"
Save Error Window As "main.pch results"
```

Select

Purpose Selects an object from an open document in the CodeWarrior Editor.

```
Select reference
```

Description The parameter *reference* is the object to select.

Listing B.11 Example for Select

```
Select text from character 5 to character 10 ¬
of line 8 of document 1
```

Set Modification Date

Purpose Sets the modification date of the specified file(s).

```
Set Modification Date filename-list to date
```

Returns A list of results of type short integer, or, if the `ExternalEditor` option is specified, a list of records of type `'ErrM'`.

Building Events

Here are the events discussed in this section:

- Bring Up to Date
- Build
- Check File Syntax
- Check Syntax
- Compile
- Compile File
- Disassemble File
- Make Project
- Precompile
- Preprocess
- Remove Binaries
- Remove Object Code
- Run
- Run Project

- Touch
- Update Project

Bring Up to Date

Purpose Brings the current target or project up to date.

```
Bring Up to Date
```

Description Performs the **Bring Up To Date** command in the Project Menu.

Returns By default, this command returns nothing.

Build

Purpose Builds the current target or project.

```
Build
```

Description Performs the **Make** command in the Project Menu.

Returns By default, `Build` returns nothing.

Check File Syntax

Purpose Checks the syntax of the specified file(s).

```
Check File Syntax filename-list
```

Description This event is equivalent to performing the **Check Syntax** command in the Project Menu. Replace *filename-list* with a single filename or a list of filenames in the project.

By default, `Check File Syntax` returns a list of short integer result codes for each file checked. A result code can either be the value of an OSErr (Operating System Error) or one of the following values:

- `noerr`

- `errShell_FileNotFound`
- `errShell_CompileError`
- `errShell_NoOpenProject`

Returns A list of results of type short integer.

Listing B.12 Examples for Check File Syntax

```
Check File Syntax "MyFile.c"
Check File Syntax {"MyFile.c", "YourFile.c"} with ExternalEditor
```

Check Syntax

Purpose Checks the syntax of the specified file(s).

```
Check Syntax filename-list [with ExternalEditor]
```

Description This event is equivalent to performing the **Check Syntax** command in the Project Menu. Replace *filename-list* with a single filename or a list of filenames in the project.

By default, `Check Syntax` returns a list of short integer result codes for each file checked. A result code can either be the value of an OSErr (Operating System Error) or one of the following values:

- `noerr`
- `errShell_FileNotFound`
- `errShell_CompileError`
- `errShell_NoOpenProject`

If the `ExternalEditor` option is used, the environment returns the Message window contents instead of the usual list of short integer results. The AppleEvent keyword for `ExternalEditor` is `'Errs'`. It takes a boolean parameter.

Returns A list of results of type short integer, or, if the `ExternalEditor` option is specified, a list of records of type `'ErrM'`.

Listing B.13 Examples for Check Syntax

```
Check Syntax "MyFile.c"
Check Syntax {"MyFile.c", "YourFile.c"} with ExternalEditor
```

Compile

Purpose Compiles the specified file(s).

```
Compile filename-list [with ExternalEditor]
```

Description This event is equivalent to performing the **Compile** command on the Project Menu. Replace *filename-list* with a single filename or a list of filenames.

By default, `Compile` returns a list of short integer result codes for each compiled file. A result code can be an OSErr value (Operating System Error) or one of the following:

- `noerr`
- `errShell_FileNotFound`
- `errShell_CompileError`
- `errShell_NoOpenProject`

If the `ExternalEditor` option is specified, the environment returns the Message window contents as a list of `'ErrM'` objects.

Returns A list of errors of type short integer or, if `ExternalEditor` is specified, of type `'ErrM'`.

Listing B.14 Examples for Compile

```
Compile "MyFile.c"
Compile {"MyFile.c", "YourFile.c"}
```

Compile File

Purpose Compiles the specified file(s).

`Compile File` *filename-list*

Description This event is equivalent to performing the **Compile** command on the Project Menu. Replace *filename-list* with a single filename or a list of filenames.

By default, `Compile` returns a list of short integer result codes for each compiled file. A result code can be an OSErr value (Operating System Error) or one of the following:

- `noerr`
- `errShell_FileNotFound`
- `errShell_CompileError`
- `errShell_NoOpenProject`

Returns A list of errors of type short integer.

Listing B.15 Examples for Compile

```
Compile File "MyFile.c"
Compile File {"MyFile.c", "YourFile.c"}
```

Disassemble File

Purpose Disassembles the specified file(s).

`Disassemble File` *filename-list*

Description This event is equivalent to performing the **Disassemble** command on the Project Menu. Replace *filename-list* with a single filename or a list of filenames.

Returns A list of errors of type short integer.

Listing B.16 Examples for Disassemble File

```
Disassemble File "MyFile.c"
Disassemble File {"MyFile.c", "YourFile.c"}
```

Make Project

Purpose Makes the current project.

```
Make Project [with ExternalEditor]
```

Description Performs the **Make** command in the Project Menu.

If the `ExternalEditor` option is used, the environment returns the Message window contents.

Returns By default, `Make Project` returns nothing. If `ExternalEditor` is specified, `Make Project` returns a list of errors of type `'ErrM'`.

Precompile

Purpose Precompiles the specified file.

```
Precompile source saving as destination [ with ExternalEditor ]
```

Description This event is equivalent to the **Precompile** command in the Project Menu. Replace *source* with the name of a file to precompile. Replace *destination* with the filename of the precompiled header.

If the `ExternalEditor` option is used, the environment returns the Message window contents.

Returns By default, `Precompile` returns nothing. If `ExternalEditor` is specified, `Precompile` returns a list of errors of type `'ErrM'`.

Listing B.17 Example for Precompile

```
Precompile "MyHeaders.pch" saving as "MyHeaders.mch"
Precompile "tip.pch" saving as "tip.mch" with ExternalEditor
```

Preprocess

Purpose Preprocesses the specified file.

```
Preprocess source[ with ExternalEditor ]
```

Description This event is equivalent to the **Preprocess** command in the Project Menu. Replace *source* with the name of a file to preprocess.

If the `ExternalEditor` option is used, the environment returns the Message window contents.

Returns By default, `Preprocess` returns nothing. If `ExternalEditor` is specified, `Preprocess` returns a list of errors of type `'ErrM'`.

Listing B.18 Examples for Preprocess

```
Preprocess "MyHeaders.c"
Preprocess "tip.c" with ExternalEditor
```

Remove Binaries

Purpose Removes the binary object code from the current project.

```
Remove Binaries
```

Description Performs the equivalent of the **Remove Object Code** command in the Project Menu.

Returns None.

Remove Object Code

Purpose Removes the binary object code from the current target or project.

```
Remove Object Code
```

Description Performs the equivalent of the **Remove Object Code** command in the Project Menu.

Returns None.

Run

Purpose Runs the current project or target.

```
Run
```

Description Performs the equivalent of the **Run** command in the Project Menu. This event builds then executes the current target if there are no compile or link errors.

Returns By default, `Run Project` returns nothing.

Run Project

Purpose Runs the current project

```
Run Project [ with ExternalEditor ] [ with SourceDebugger ]
```

Description Performs the equivalent of the **Run** command in the Project Menu. This event builds then executes the current project if there are no compile or link errors.

If the `ExternalEditor` option is used, the environment returns the Message window contents.

If the `SourceDebugger` option is used, the environment launches the successfully-built project into the source-level debugger.

Returns By default, `Run Project` returns nothing. If `ExternalEditor` is specified, `Run Project` returns a list of errors that occurred when running the project, of type `'ErrM'`.

Listing B.19 Examples for Run Project

```
Run Project with SourceDebugger
Run Project with ExternalEditor
```

Touch

Purpose Touches the specified file(s).

```
Touch filename
```

Description Performs the equivalent of clicking the Touch column in a Project window. Touching a file forces it to be recompiled during a make operation. Replace *filename* with a single file or a list of files to touch.

For more on touching a file to be recompiled, consult "Synchronizing modification dates" on page 84.

Returns A list of errors. Each result code can have one of the following values:

- `noErr`
- `errShell_FileNotFound`
- `errShell_NoOpenProject`

Listing B.20 Examples for Touch

```
Touch "MyFile.c"
Touch { "MyFile.c", "YourFile.c" }
```

Update Project

Purpose Updates the current project.

```
Update Project [ with ExternalEditor ]
```

Description This command is equivalent to the **Bring Up To Date** command in the Project Menu. If the `ExternalEditor` option is used, the environment returns the Message window contents.

Returns By default, `Update Project` returns nothing. If `ExternalEditor` is specified, `Update Project` returns a list of errors of type `'ErrM'`.

Status/Query Events

Here are the events discussed in this section:

- Get
- Set
- Get Definition
- Get Member Function Names
- Get Nonsimple Classes
- Get Open Documents
- Get Preferences
- Set Preferences
- Get Project File
- Set Project File
- Set Current Target
- Set Default Project
- Get Project Specifier
- Get Segments
- Set Segment
- Is In Project
- Reset File Paths
- Close

- Count
- Make

Get

Purpose Gets the object referenced.

```
Get reference [ as list of typeclass ]
```

Description The parameter *reference* is the object whose data is to be returned. The parameter *typeclass* is the desired types for the data, in order of preference.

Set

Purpose Sets the object referenced.

```
Set reference to anything
```

Description The parameter *reference* is the object whose data is to be changed. The parameter *anything* is the new value for the object.

Listing B.21 Example for Set

```
tell application "CodeWarrior IDE 3.2" to
set numClasses to the count of classes
```

Get Definition

Purpose Queries the location(s) of a globally-scoped function or data object for the current project.

```
Get Definition string
```

Description The *string* is the name of the symbol you are interested in.

Returns Record containing a list of the function information.

Listing B.22 Example for Get Definition

```
Get Definition "main"
```

Get Member Function Names

Purpose Gets a list of all the member functions of a class object.

```
Get Member Function Names reference
```

Returns List containing the information.

Listing B.23 Example for Get Member Function Names

```
Get Member Function Names class "CPowerTelnetApp"
```

Get Nonsimple Classes

Purpose Gets a list of all the member functions of a class object.

```
Get Nonsimple Classes
```

Returns List containing the information.

Listing B.24 Example for Get Nonsimple Classes

```
Get Nonsimple Classes class "CPowerTelnetApp"
```

Get Open Documents

Purpose Gets the list of open documents.

```
Get Open Documents
```

Returns List of documents in records of type `'docu'`. See Table B.40 for more information

Get Preferences

Purpose Gets settings from a panel.

```
Get Preferences [ of pref-list ] from panel panel-name
```

Description The *panel-name* must be the name of the preference panel file and not the name that appears in the preferences window. For example, to set C/C++ Language options, use `"C/C++ Compiler"` as the *panel-name* and not `"C/C++ Language"`.

Returns Record containing a list of the requested preferences. If you do not include *pref-list*, it returns all the preferences for *panel-name*.

Listing B.25 Examples for Get Preferences

```
Get Preferences from panel "C/C++ Compiler"
Get Preferences of {File Name, SIZE Flags} ¬
  from panel "PPC Project"
```

Set Preferences

Purpose Specifies the settings for a panel.

```
Set Preferences of panel panel-name to record
```

Description Performs the equivalent of setting options using either the Preferences or Settings windows. This event lets you set the properties of the current project. It is not necessary to specify every preference, those not mentioned in the record retain their settings. The properties for different panels are listed in various tables from Table B.3 to Table B.38.

The *panel-name* must be the name of the preference panel file and not the name that appears in the preferences window. For example, to set C/C++ Language options, set *panel-name* to `"C/C++ Compiler"` and not `"C/C++ Language"`.

Returns None.

Listing B.26 Examples for Set Preferences

```
Set Preferences of panel "PPC Project" to { ¬
  File Name:"MyProgram", File Creator:"Mine", SIZE Flags:23008 ¬
}
Set Preferences of panel "C/C++ Compiler" to { ¬
  Prefix File: "MacHeaders", ¬
  Activate CPlusPlus: TRUE, ¬
  Require Function Prototypes: FALSE ¬
}
Set Preferences of panel "C/C++ Warnings" to { ¬
  Extended Error Checking: TRUE ¬
}
```

Get Project File

Purpose Gets information on a project entry.

```
Get Project File file-number segment seg-number
```

Returns The information of the specified entry in the current project as a record of type `'SrcF'`. The *file-number* parameter specifies a file within its segment or group. The *seg-number* parameter specifies a segment or group within the project. Numbering for both parameters are short integers beginning at 1.

Listing B.27 Examples for Get Project File

```
get project file 1 segment 1
    -- First entry in project
get project file 1 segment 2
    -- First entry in 2nd segment
```

Set Project File

Purpose Sets information on a project entry.

```
Set Project File filename to record
```

Description Changes the settings for the specified entry in the open project. The *record* parameter is of type `'SrcF'`. Only the symbols and weak link fields are allowed. Listing B.28 shows how to set weak linking for a library `InterfaceLib` that is in a project.

Returns None.

Listing B.28 Example for Set Project File

```
Set Project File "InterfaceLib" to {weak link: true}
```

Set Current Target

Purpose Sets the current target for a project.

```
Set Current Target name-of-target
```

Description This AppleEvent causes the build target to change. This event would be useful when changing the build target in a project, so that a new target can be built or otherwise operated on with other AppleEvents. To learn more information about setting the current build target, refer to "Set Current Target" on page 395.

Returns None.

Listing B.29 Example for Set Current Target

```
Set Current Target "Muscle 68K"
```

Set Default Project

Purpose Sets the default project.

```
Set Default Project name-of-target
```

Description This AppleEvent causes the default project to change. To learn more information about setting a default project, refer to "Set Default Project" on page 395.

Returns None.

Listing B.30 Example for Set Default Project

```
Set Current Project "Muscle.µ"
```

Get Project Specifier

Purpose Gets the filename of the project.

```
Get Project Specifier
```

Returns The name of the current project.

Get Segments

Purpose Gets the descriptions of all segments/groups in the open project.

```
Get Segments
```

Returns List of documents in records of type `'Seg '`. Refer to "Segment" on page 475 for more information

Set Segment

Purpose Sets preferences for the current project.

```
Set Segment number to record
```

Description Sets information for a segment or group in the open project. Segment numbering starts at 1. The *record* parameter is an object of type `'Seg '`. Listing B.31 shows how to rename a segment/group in a project. Refer to "Segment" on page 475 for more information.

Returns None.

Listing B.31 Example for Set Segment

```
Set Segment 1 to {name:"New Sources"}
```

Is In Project

Purpose Are the specified file(s) are in the project?

```
Is In Project filename
```

Description Replaces *filename* with a single filename or a list of filenames.

Returns A list of errors. The result code for each specified file can have the following values:

- `noErr` if the file is in the project
- `errShell_FileNotFound` if file is not in the project.

Listing B.32 Examples for Is In Project

```
Is In Project "SillyBalls.c"
Is In Project { "SillyBalls.c", "Initialize.c" }
```

Reset File Paths

Purpose Resets access paths for all files belonging to the open project.

```
Reset File Paths
```

Returns None.

Close

Purpose Closes an object. The `saving` and `saving in` parameters are optional, and have the range of values listed here.

```
Close reference [ saving yes/no/ask ] [ saving in alias ]
```

Returns None.

Count

Purpose Counts the number of elements within an object.

```
Count reference each type class
```

Returns An integer indicating the number of elements.

Make

Purpose Makes a new element. The `as list of`, `as`, `with data`, and `with properties` parameters are optional, and have the range of values listed here.

```
Make new [ as list of type class ] [ at location reference ] [
with data anything ] [ with properties record ]
```

Returns A reference to the new object(s).

Navigation Events

Here are the events discussed in this section:

- Goto Function
- Goto Line
- Open Browser

Goto Function

Purpose Jumps to the specified function defined in the active editor window.

```
Goto Function name
```

Description This event is equivalent to selecting a routine name in the active editor window's function pop-up menu. The insertion point does not move when a function is selected with this event.

Returns None.

Listing B.33 Examples for Goto Function

```
Goto Function "main"
Goto Function "SkipBlanks"
```

Goto Line

Purpose Jumps to the specified line number in the active editor window.

```
Goto Line number
```

Description `Goto Line` moves the insertion point to the specified line number, *number*, in the active editor window. If the line number specified exceeds the last line number, the insertion point is placed at the last line.

Returns None.

Listing B.34 Examples for Goto Line

```
Goto Line 1
Goto Line 493
```

Open Browser

Purpose Displays a class, member function, or data member object in a single class browser window. You cannot display a procedural function.

```
Open Browser reference
```

Returns None.

Listing B.35 Example for Open Browser

```
Open Browser class "CPowerTelnetApp"
Open Browser member function 2 of class "CPowerTelnetApp"
```

CodeWarrior IDE AppleScript Classes

CodeWarrior events have several classes to let you control the CodeWarrior IDE's actions and option settings. These classes are based on the items available in the Preferences and Target Settings windows.

The available classes are determined by the compiler (C/C++, Java, or Pascal) and the processor for which it is generating object code (68k-based, Intel x86 or PowerPC-based Macintosh). In other words, the preference classes available for the CodeWarrior C/C++ compiler that generates object code for the PowerPC are different from those available for the CodeWarrior C compiler that generates 68K object code.

AppleScript classes for the CodeWarrior environment are, for the most part, separated by preference panels, project settings, and then by miscellaneous environment items.

- Project Classes—contains properties for each aspect of a project, from target parameters to final application settings.
- Compiler Classes—contains properties for each language and compiler, to configure compilation settings and warnings.
- CodeGen Classes—contains properties for each possible kind of code that can be generated by the CodeWarrior compilers.
- Disassembler Classes—contains properties for the disassemblers.
- Linker Classes—contains properties for the linker settings.
- Build Classes—contains properties for build environment settings and errors.
- Browser Classes—contains properties for the code browser.

- Editor Classes—contains properties for the CodeWarrior IDE text editor.
- Object Classes—contains properties that describe objects, including data members, classes and base classes, and member functions.
- Misc Classes—contains properties that describe miscellaneous aspects of the CodeWarrior IDE environment.

Many options that use a pop-up menu in the preference panel now use an enumerated type to specify their values, instead of an integer. For example, to set the Code Model, you should now use small, smart, or large, instead of 1, 2, or 3.

WARNING! Your script will not work if you use integer to set a property that expects a symbol.

Project Classes

- 68K Project
- PowerPC Project
- Java Project
- Win32/x86 Project
- Access Paths
- Path Information
- Target Settings
- File Mapping Information
- Segment
- Project File

68K Project

Table B.3 lists the 68K Project class properties.

Table B.3 68K Project Class

Name	Property Type
Project Type	• standard application • CFM68K application • code resource • library • shared library • MPW Tool • Pilot Application • Pilot Code Resource
File Name	string
File Creator	string
File Type	string
Minimum Size	integer
Preferred Size	integer
SIZE Flags	small integer (SIZE flag bits must be computed as an integer value)
SYM File	string
Resource Name	string
Display Dialogs	boolean
Merge To File	boolean
Resource Flags	small integer
Resource Type	string
Resource ID	small integer
Multi Segment	boolean

Name	Property Type
Library Type	• A4 relative • A5 relative • CFM68K • Pilot Library
Seg Type	string
Stack Size	integer
Start-up Code	• standard • The Debugger Aware • custom
Header Type	• standard • device driver • desk accessory • custom
RSEG Application	boolean

PowerPC Project

Table B.4 lists the PPC Project class properties.

Table B.4 PPC Project Class

Name	Property Type
Project Type	• standard application • code resource • library • shared library
File Name	string
File Creator	string
File Type	string

Name	Property Type
Minimum Size	integer
Preferred Size	integer
SIZE Flags	small integer (SIZE flag bits must be computed as an integer value)
SYM File	string
Resource Name	string
Display Dialogs	boolean
Merge To File	boolean
Resource Flags	small integer
Resource Type	string
Resource ID	small integer
Stack Size	integer
Header Type	• none • native

Java Project

Table B.5 lists the Java Project class properties.

Table B.5 Java Project Class

Name	Property Type
Main Class	string
Java Project Type	• java applet • java application • java library
Arguments	string

Name	Property Type
`Compress`	`boolean`
`HTML Helper App`	`string`

Win32/x86 Project

Table B.6 lists the x86 Project class properties.

Table B.6 x86 Project Class

Name	Property Type
`Project Type`	• `standard application` • `shared library` • `library`
`File Name`	`string`
`Min Heap Size`	`integer`
`Preferred Heap Size`	`integer`
`Base Address`	`integer`
`Max Stack Size`	`integer`
`Min Stack Size`	`integer`

Access Paths

Table B.7 lists the properties for the Access Paths Class.

Table B.7 Access Paths Class

Name	Property Type
`User Paths`	`list` (of path information records)
`Always Full Search`	`boolean`

Name	Property Type
`Convert Paths`	`boolean`
`System Paths`	`list` (of path information records)

Path Information

A path information record may contain the properties shown in Table B.8. It must contain at least the name field.

Table B.8 Path Information Class

Name	Property Type
`name`	`string`
`recursive`	`boolean`
`origin`	• `absolute` • `project relative` • `shell relative` • `system relative`
`host flags`	• `1` (for the Mac OS) • `2` (for Windows)

If you use a string instead of a path information record, CodeWarrior sets recursive to true and origin to project relative.

To clear all the access path entries listed in the Access Paths preference panel, set the User Paths or System Paths property to an empty list. For example, in AppleScript, this statement removes all entries, including the default entries, `{Project` *`f`*`}` and `{Compiler` *`f`*`}`:

```
Set Preferences of panel "Access Paths" to ¬
  {User Paths: {}, System Paths: {} }
```

To add a default entry back, add an access path record with the name set to ":" and with the origin set to project relative (for

{Project *f*}) or shell relative (for {Compiler *f*}). For example, this statement sets User Paths to {Project *f*} and System Paths to {Compiler *f*}:

```
Set Preferences of panel "Access Paths" to ¬
  {User Paths:   {{name: ":", ¬
                  origin: project relative}},¬
   System Paths: {{name: ":", ¬
                  origin: shell relative}}}
```

For more information on the meaning of the properties listed in Table B.7 and Table B.8, see "Access Paths" on page 298.

Target Settings

You set the current target parameters in the Target Settings options panel with the properties shown in Table B.9.

Table B.9 Target Settings Class

Name	Property Type
Target Name	string
Linker	string
Post Linker	string
Output Directory Path	string
Output Directory Origin	• absolute • project relative • shell relative • system relative
Pre Linker	string

The Target Name string is the name of the target (you choose this name). The Linker string must be the name of one of the files in the Linkers folder of the CodeWarrior Plugins folder. The Post

`Linker` string must be the name of one of the files in the `Post Linkers` folder of the `CodeWarrior Plugins` folder. The `Output Directory Path` string is a string that points to a location on your hard disk where the output files should be placed after linking. You can make this path absolute, or you can make it relative to the location of the project (project relative), compiler (compiler relative), or system (system relative).

The following is a list of the names of the linkers included with CodeWarrior:

- `MacOS 68K Linker`
- `MacOS PPC Linker`
- `MacOS Merge`
- `Java Linker`
- `MW JavaDoc Linker`
- `Win32 x86 Linker`

For example, the following statement changes the current target to Macintosh 68K:

```
Set Preferences of panel "Target Settings" to ¬
  {Linker: "MacOS 68K Linker"}
```

File Mapping Information

The File Mapping Information is a list of all the types of files you can include in the current project. It contains records described in Table B.10.

Table B.10 File Mapping Information Class

Name	Property Type
`File Type`	`string`
`Extension`	`string`
`Precompiled`	`boolean`

Name	Property Type
`Resource File`	`boolean`
`Launchable`	`boolean`
`Ignored by Make`	`boolean`
`Compiler`	`string`

The `Compiler` string must be the name of one of the files in the `Compilers` folder of the `CodeWarrior Plugins` folder. These names are different from the names that appear in the Compiler pop-up menu of the Target preference panel. Table B.11 shows you which name to use for the compilers included with CodeWarrior.

Table B.11 Choosing a compiler

To target...	with...	specify this string.
68K Macintosh	Metrowerks C/C++	"MW C/C++ 68K"
	Metrowerks Pascal	"MW Pascal 68K"
	Rez	"Rez"
	Library Importer	"Lib Import 68K"
	MPW .o Importer	"MPW Import 68K"
	PEF Importer	"PEF Import 68K"
Power Macintosh	Metrowerks C/C++	"MW C/C++ PPC"
	Metrowerks Pascal	"MW Pascal PPC"
	Rez	"Rez"
	Library Importer	"Lib Import PPC"
	PEF Importer	"PEF Import PPC"
	XCOFF Importer	"XCOFF Import PPC"

To target...	with...	specify this string.
`Win32/x86`	`Metrowerks C/C++`	`"MW C/C++ x86"`
	`Resource Compiler`	`"MW WinRC"`
	`Resource Importer`	`"WinRes Import"`
	`x86 Lib Importer`	`"Lib Import x86"`
	`x86 Obj Import`	`"Obj Import x86"`
`Java`	`Java`	`"MW Java"`

To specify that a file isn't compiled, use the empty string `""` for the compiler. For example, these statements show how to add an entry for text files that end in `.txt` and are not compiled.

```
set currPrefs to Get Preferences from panel "Target"
set Mappings of currPrefs to ¬
  Mappings of currPrefs & ¬
  {{File Type:"TEXT", Extension:".txt", ¬
    Compiler:"", Precompiled:false, ¬
    Resource File:false, Launchable:true, ¬
    Ignored by Make:true}}
Set Preferences of panel "Target" to currPrefs
```

Segment

The Segment Class properties, as shown in Table B.12, contain information about a segment or group in the open project,

Table B.12 Segment Class

Name	Property Type
`name`	`string`
`filecount` (read only)	`small integer`

Name	Property Type
`seg-preloaded` (68K only)	`boolean`
`seg-protected` (68K only)	`boolean`
`seg-locked` (68K only)	`boolean`
`seg-purgeable` (68K only)	`boolean`
`seg-system heap` (68K only)	`boolean`

Project File

The Project File Class contains information about an entry in a project file. Table B.13 illustrates the available properties.

Table B.13 Project File Class

Name	Property Type
`filetype` (read only)	• `source` • `unknown`
`name` (read only)	`string`
`disk file` (read only)	`file specification`
`codesize` (read only)	`integer`
`datasize` (read only)	`integer`
`up to date` (read only)	`boolean`
`symbols`	`boolean`
`initialize before`	`boolean`

Name	Property Type
`includes` (read only)	`file specification`
`weak link` (PPC only)	`boolean`

Compiler Classes

- C/C++ Compiler
- Java Compiler
- Pascal Compiler
- Rez Resource Compiler
- Windows Resource Compiler

C/C++ Compiler

Table B.14 lists the CodeWarrior C/C++ Compiler class properties.

NOTE: In AppleScript, you must refer to the C/C++ Language preference panel as: `panel "C/C++ Compiler"`.

Table B.14 C/C++ Compiler Class

Name	Property Type
`Prefix File`	`string`
`Activate CPlusPlus`	`boolean`
`ARM Conformance`	`boolean`
`ANSI Keywords Only`	`boolean`
`Require Function Prototypes`	`boolean`
`Expand Trigraph Sequences`	`boolean`
`Enums Always Ints`	`boolean`
`MPW Pointer Type Rules`	`boolean`

Name	Property Type
Exception Handling	boolean
AutoInlining	boolean
Pool Strings	boolean
Dont Reuse Strings	boolean
ANSI Strict	boolean
MPW Newlines	boolean
RTTI	boolean
Multibyte Aware	boolean
Enable wchar_t	boolean
Use Unsigned Chars	boolean
ECPlusPlus Compatibility	boolean
Objective C	boolean
Inlining	• inline_none • inline_smart • inlinedepth_1 • inlinedepth_2 • inlinedepth_3 • inlinedepth_4 • inlinedepth_5 • inlinedepth_6 • inlinedepth_7 • inlinedepth_8 • inline_always
Enable bool Support	boolean

Name	Property Type
`Direct To SOM`	• `SOMoff` • `SOMon` • `SOMonWithEnv`
`Deferred Inlining`	• `boolean`

Table B.15 lists the CodeWarrior C/C++ Warnings class properties.

Table B.15 C/C++ Warnings Class

Name	Property Type
`Unused Variables`	`boolean`
`Inconsistent Class Struct`	`boolean`
`Unused Arguments`	`boolean`
`Illegal Pragmas`	`boolean`
`Empty Declarations`	`boolean`
`Possible Errors`	`boolean`
`Extra Commas`	`boolean`
`Extended Error Checking`	`boolean`
`Treat Warnings As Errors`	`boolean`
`Hidden Virtual Functions`	`boolean`
`Implicit Arithmetic` Conversions	`boolean`
`NonInlined Functions`	`boolean`

Java Compiler

Table B.16 lists the CodeWarrior Java Compiler class properties.

Table B.16 Java Compiler Class

Name	Property Type
`Method Inlining`	`boolean`

Pascal Compiler

Table B.17 lists the Pascal Language Class options. (In AppleScript, you must refer to the Pascal Language preference panel as: `panel "Pascal Compiler"`)

Table B.17 Pascal Compiler Class

Name	Property Type
`Activate Range Checking`	`boolean`
`Use Propagation`	`boolean`
`Activate Overflow Checking`	`boolean`
`Case Sensitive`	`boolean`
`ANS Conformance`	`boolean`
`Activate ObjectPascal`	`boolean`
`Strings copy using length byte`	`boolean`
`Pool Strings`	`boolean`
`Dont Reuse Strings`	`boolean`
`Pool Sets`	`boolean`
`Dont Reuse Sets`	`boolean`
`Prefix File`	`string`
`Relax Pointer Compatibility`	`boolean`
`Optimize class hierarchy`	`boolean`
`Pointer based objects`	`boolean`

Name	Property Type
Expand method tables	boolean
Inline method dispatching	boolean
Activate NilChecking	boolean
Trap Unmatched Cases	boolean
Copy Value Parameter	boolean
Turbo Pascal IO	small integer

Table B.18 lists the Pascal Warnings class properties.

Table B.18 Pascal Warnings Class

Name	Property Type
Modified ForLoop Indexes	boolean
Function Returns	boolean
Undefined Routines	boolean
GotoAndLabels	boolean
BranchingIntoWith	boolean
BranchingIntoFor	boolean
BranchingBetweenCase	boolean
BranchingBetweenIfAndElse	boolean
Unused Variables	boolean
Unused Arguments	boolean
Check string param sizes	boolean

Rez Resource Compiler

Table B.19 lists the properties for the CodeWarrior Rez Compiler Class.

Table B.19 Rez Compiler Class

Name	Property Type
`Redeclared Types`	`boolean`
`RezPrefix File`	`string`
`Escape Control Chars`	`boolean`
`Max width`	`small integer`
`Filter Mode`	`Skip`, or `Only`
`Filtered Types`	`string`
`Alignment`	`small integer`
`Script Mode`	`Roman`, `Japanese`, `Korean`, `SimpChinese`, or `TradChinese`

Windows Resource Compiler

Table B.20 lists the properties for the Windows Resource Compiler Class.

Table B.20 Windows Resource Compiler Class

Name	Property Type
`Prefix File`	`string`

CodeGen Classes

- 68K CodeGen
- PPC CodeGen
- IR Optimizer
- Win32/x86 CodeGen

68K CodeGen

Table B.21 lists the 68K Processor class properties. In AppleScript, you must refer to the 68K Processor preference panel as: `panel "68K CodeGen"`

Table B.21 68K CodeGen Class

Name	Property Type
`Struct Alignment`	• `Align_68k` • `Align_68k_4byte` • `Align_PPC`
`Peephole Optimizer`	`boolean`
`CSE Optimizer`	`boolean`
`Optimize For Size`	`boolean`
`Use Profiler`	`boolean`
`Code Model`	• `small` • `smart` • `large`
`MC68020 CodeGen`	`boolean`
`Floating Point CodeGen`	• `SANE` • `MC68881` • `Library` • `PalmOS`
`Far Method Tables`	`boolean`
`Far String Constants`	`boolean`
`Four Bytes Ints`	`boolean`
`Eight Byte Double`	`boolean`
`Far Data`	`boolean`

Name	Property Type
PC Relative Strings	boolean
MPW Calling Conventions	boolean

PPC CodeGen

Table B.22 lists the PPC Processor preference panel class properties. In AppleScript, you must refer to the PPC Processor preference panel as: `panel "PPC CodeGen"`

Table B.22 PPC CodeGen Class

Name	Property Type
Struct Alignment	• Align_68k • Align_68k_4byte • Align_PPC
Processor	• PPC_Generic • PPC_601 • PPC_603 • PPC_603e • PPC_604 • PPC_604e • PPC_750
Peephole Optimizer	boolean
Use Profiler	boolean
Make String ReadOnly	boolean
Schedule	boolean
Store Data in TOC	boolean
Use FMADD Instructions	boolean
Processor Specific	boolean

Name	Property Type
`Traceback Tables`	• `TB_None` • `TB_Inline` • `TB_OutOfLine`
`Altivec`	• `boolean`

IR Optimizer

Table B.23 lists the IR Optimizer class properties.

Table B.23 IR Optimizer Class

Name	Property Type
`Optimize Space`	`boolean`
`Optimize Speed`	`boolean`
`Common Subexpressions`	`boolean`
`Loop Invariants`	`boolean`
`Propagation`	`boolean`
`Dead Store Elimination`	`boolean`
`Strength Reduction`	`boolean`
`Dead Code Elimination`	`boolean`
`Lifetime Analysis`	`boolean`
`Optimizations Log`	`boolean`

Win32/x86 CodeGen

Table B.24 lists the x86 CodeGen class properties.

Table B.24 Win32/x86 CodeGen Class

Name	Property Type
Peephole Optimizer	boolean
Machine Code Listing	boolean
Byte Alignment	small integer
Sym Debug Information	boolean
CodeView Debug Info	boolean
Register Coloring	boolean
Expand Intrinsics	boolean
Disable Optimizations	boolean
Instruction Scheduling	boolean
Target Processor	• Generic X86 • Pentium • Pentium Pro • Pentium II • AMD K6
Instruction Set	• None • MMX • MMX_K63D • K6 3D

Disassembler Classes

- 68K Disassembler
- PowerPC Disassembler

68K Disassembler

Table B.25 lists the properties for the 68K Disassembly Class.

Table B.25 68K Disassembler Class

Name	Property Type
Show Code	boolean
Show Source	boolean
Dont show hex	boolean
Show Data	boolean
Show Exceptions	boolean
Show SYM	boolean
Show Names	boolean

PowerPC Disassembler

Table B.26 lists the properties for the PowerPC Disassembly Class.

Table B.26 PowerPC Disassembly Class

Name	Property Type
Show Code	boolean
Show Source	boolean
Dont show hex	boolean
Show Data	boolean
Show Exceptions	boolean
Show SYM	boolean
Show Names	boolean
Use Extended Mnemonics	boolean

Linker Classes

- 68K Linker
- CFM68K Linker

- Java Linker
- Mac OS Merge Linker
- PowerPC Linker
- PowerPC PEF Linker
- Win32/x86 Linker

68K Linker

Table B.27 lists the 68K Linker class properties.

Table B.27 68K Linker Class

Name	Property Type
`Generate SYM File`	`boolean`
`Full Path In Sym Files`	`boolean`
`Generate Link Map`	`boolean`
`Fast Link`	`boolean`
`Suppress Warnings`	`boolean`
`MacsBug Symbols`	• `none` • `oldsymbols` • `newsymbols`
`Generate A6 Stack Frames`	`boolean`
`Link Single Segment`	`boolean`
`Merge Compiler Glue`	`boolean`
`Strip Static Init Code`	`boolean`

CFM68K Linker

Table B.28 lists the CFM68K Linker class properties.

Table B.28 CFM68K Linker Class

Name	Property Type
Export Symbols	• none • expfile • all • pragma
Old Definition	integer
Old Implementation	integer
Current Version	integer
Share Data Section	boolean
Expand Uninitialized Data	boolean
Fragment Name	string
Initialization Name	string
Main Name	string
Termination Name	string
Force Indirect Access	boolean
Far Data Threshold	integer
Global Data Alignment	• align1byte • align2byte • align4byte • align8byte
Library Folder ID	small integer

Java Linker

Table B.29 lists the Java Linker class properties.

Table B.29 Java Linker Class

Name	Property Type
`File Name`	`string`
`File Creator`	`string` Possible values include '`JAVA`' or '`MWZP`' as well as the creator types for Metrowerks Java or ClassWrangler.
`File Type`	`string` (this is 4 characters: '`ZIP `')
`Output Type`	• `zip file` • `runable zip file` • `droplet` • `folder`
`Compress Zip`	`boolean`

Mac OS Merge Linker

Table B.30 lists the Mac OS Merge Linker class properties.

Table B.30 Mac OS Merge Linker Class

Name	Property Type
`Project Type`	`constant` (The type of the project)
`File Name`	`boolean`
`File Creator`	`string` (The creator type of the finished binary)

Name	Property Type
File Type	string (The file type of the finished binary)
Suppress Warnings	boolean
Copy Fragments	boolean
Copy Resources	boolean
Skip Resource Types	string

PowerPC Linker

Table B.31 lists the PPC Linker class properties.

Table B.31 PPC Linker Class

Name	Property Type
Generate SYM File	boolean
Full Path In Sym Files	boolean
Generate Link Map	boolean
Link Mode	• fast • normal • slow
Suppress Warnings	boolean
Initialization Name	string
Main Name	string
Termination Name	string
Strip Static Init Code	boolean
Duplicate Item Warning	boolean

PowerPC PEF Linker

Table B.32 lists the PPC PEF class properties.

Table B.32 PPC PEF Class

Name	Property Type
`Export Symbols`	• `none` • `expfile` • `all` • `pragma`
`Old Definition`	`integer`
`Old Implementation`	`integer`
`Current Version`	`integer`
`Code Sorting`	• `nosort` • `pragmas` • `depth` • `breadth` • `sortfile`
`Share Data Section`	`boolean`
`Expand Uninitialized Data`	`boolean`
`Fragment Name`	`string`
`Library Folder ID`	`small integer`
`Collapse Reloads`	`boolean`

Win32/x86 Linker

Table B.33 lists the x86 Linker class properties.

Table B.33 x86 Linker Class

Name	Property Type
Generate SYM File	boolean
Entry Point Usage	• none • default • user specified
Entry Point	string
SubSystem	• unknown • native • Windows GUI • Windows CUI
SubSystem Major Id	small integer
SubSystem Minor Id	small integer
User Major Id	small integer
User Minor Id	small integer
Generate Link Map	boolean
Generate CV Info	boolean
Command Line File	string

Build Classes

- Build Extras
- Error Information

Build Extras

Table B.34 describes the properties for the Build Extras Class.

Table B.34 Build Extras Class

Name	Property Type
`Browser active`	`boolean`
`Modification date caching`	`boolean`
`Dump Browser Info` (read only)	`boolean`
`Cache Subproject Data` (read only)	`boolean`

Error Information

This class describes a single error or warning from the compiler or the linker. This class is used by all compilers for all processors. The properties for this class are listed in Table B.35.

Table B.35 Error Information Class

Name	Property Type
`messageKind` (read only)	• `information` • `compiler error` • `compiler warning` • `definition` • `linker error` • `linker warning` • `find result` • `generic error`
`message` (read only)	`string`
`disk file` (read only)	`file specification`
`line Number` (read only)	`integer`

Browser Classes

- Browser Coloring
- Browser Catalog
- Function Information

Browser Coloring

Table B.36 lists the Browser Coloring preference panel properties.

Table B.36 Browser Coloring Class

Name	Property Type
`Browser Keywords`	`boolean`
`Classes Color`	`RGB values list`
`Constants Color`	`RGB values list`
`Enums Color`	`RGB values list`
`Functions Color`	`RGB values list`
`Globals Color`	`RGB values list`
`Macros Color`	`RGB values list`
`Templates Color`	`RGB values list`
`Typedefs Color`	`RGB values list`

Browser Catalog

The Browser Catalog Class elements may be referred to by numeric index, and by name.

Function Information

The Function Information class properties are described in Table B.37.

Table B.37 Function Information Class Properties

Name	Property Type
`disk file` (read only)	`file specification`
`lineNumber` (read only)	`integer`

Editor Classes

- Editor
- Font
- Document
- Character
- Insertion Point
- Custom Keywords
- Line
- Text
- Selection-Object
- Syntax Coloring
- Window

Editor

Table B.38 lists the Editor class properties.

Table B.38 Editor Class

Name	Property Type
`Remember window`	`boolean`
`Main Text Color`	`RGB values list`
`Background Color`	`RGB values list`
`Context Popup Delay`	`boolean`

Name	Property Type
Remember selection	boolean
Use Drag & Drop Editing	boolean
Flash delay	integer
Dynamic scroll	boolean
Balance	boolean
Remember font	boolean
Sort Function Popup	boolean
Use Multiple Undo	boolean
Save on update	boolean

An RGB values list is a list of three numbers from 0 to 65,535 that specifies how much red, green, and blue a color contains. For example, this example code sets the main text color to red.

```
set Prefs to Get Preferences from panel "Editor"
set Main Text Color of Prefs to {65535,0,0}
Set Preferences of panel "Editor" to Prefs
```

Font

Table B.39 lists the Font class properties.

Table B.39 Font Class

Name	Property Type
Auto Indent	boolean
Tab size	small integer
Text font	string
Text size	small integer

Document

This class, shown in Table B.40, contains class properties for a text file opened with the CodeWarrior Editor. The plural form for this class should be referred to as Documents.

The elements for a document are:

- `character` by numeric index, before/after another element, as a range of elements, or satisfying a test
- `insertion point` before/after another element
- `line` by numeric index, as a range of elements, before/after another element
- `text` as a range of elements

Table B.40 Document Class

Name	Property Type
`name` (read only)	`string`
`kind`	• `project` • `editor document` • `message` • `file compare` • `catalog document` • `class browser` • `single class browser` • `symbol browser` • `class hierarchy` • `single class hierarchy` • `project inspector` • `ToolServer worksheet` • `build progress document`

Name	Property Type
`file permissions` (read only)	• `read write` • `read only` • `checked out read write` • `checked out read only` • `checked out read modify` • `locked` • `none`
`location` (read only)	`file specification`
`index` (read only)	`integer`
`window` (read only)	`window`

Character

Table B.41 describes the Character class properties.

Table B.41 Character Class Properties

Name	Property Type
`offset` (read only)	`integer`
`length` (read only)	`integer`

Insertion Point

Table B.42 describes the Insertion Point Class properties.

Table B.42 Insertion Point Class Properties

Name	Property Type
`length` (read only)	`integer`
`offset` (read only)	`integer`

Custom Keywords

Table B.43 describes the Custom Keywords class properties.

Table B.43 Custom Keywords Class Properties

Name	Property Type
`Custom color 1`	`RGB color values list`
`Custom color 2`	`RGB color values list`
`Custom color 3`	`RGB color values list`
`Custom color 4`	`RGB color values list`

Line

Table B.44 describes the properties for the Line class. The plural form of the class should be referred to as Lines. This class has elements that may be described as follows:

- `character` by numeric index, as a range of elements, and before/after another element

Table B.44 Line Class Properties

Name	Property Type
`index` (read only)	`integer`
`offset` (read only)	`integer`
`length` (read only)	`integer`

Text

Table B.45 describes the Text class properties. The Text Class has the following elements:

- `character` by numeric index, before/after another element, as a range of elements
- `insertion point` before/after another element
- `line` by numeric index, as a range of elements, before/after another element
- `text` as a range of elements

Table B.45 Text Class Properties

Name	Property Type
`offset` (read only)	`integer`
`length` (read only)	`integer`

Selection-Object

Table B.46 describes the Selection-Object class properties. The elements of this object are:

- `character` by numeric index, before/after another element, as a range of elements, or satisfying a test
- `line` by numeric index, as a range of elements, or before/after another element
- `text` as a range of elements

Table B.46 Selection-Object Class Properties

Name	Property Type
`contents`	`type class`
`length` (read only)	`integer`
`offset` (read only)	`integer`

Syntax Coloring

Table B.48 describes the Syntax Coloring class properties.

Table B.47 Syntax Coloring Class Properties

Name	Property Type
`Syntax coloring`	`boolean`
`Comment color`	`RGB color values list`
`Keyword color`	`RGB color values list`
`String color`	`RGB color values list`
`Custom color 1`	`RGB color values list`
`Custom color 2`	`RGB color values list`
`Custom color 3`	`RGB color values list`
`Custom color 4`	`RGB color values list`

Window

Table B.48 describes the Window class properties. The plural form of this object is Windows.

Table B.48 Window Class Properties

Name	Property Type
`name`	`string`
`index`	`integer`
`bounds`	`bounding rectangle`
`document` (read only)	`document`
`position` (read only)	`point`
`visible` (read only)	`boolean`
`zoomed`	`boolean`

Object Classes

The object classes describe the properties of the objects in the project.

- Member Function Class
- Base Class
- Class Class
- Data Member Class

Member Function Class

Table B.49 lists the Member Function AppleScript class properties. The plural reference to use would be Member Functions.

Table B.49 Member Function Class

Name	Property Type
`name` (read only)	`string`
`access` (read only)	• `public` • `pro-tected` • `private`
`virtual` (read only)	`boolean`
`static` (read only)	`boolean`
`declaration file` (read only)	`file specifi-cation`
`declaration start offset` (read only)	`integer`
`declaration end offset` (read only)	`integer`
`implementation file` (read only)	`file specifi-cation`

Name	Property Type
`implementation end offset` (read only)	`integer`
`implementation start offset` (read only)	`integer`

Base Class

Table B.50 lists the Base Class AppleScript class properties. The plural reference to use would be Base Classes.

Table B.50 Base Class AppleScript Class Properties

Name	Property Type
`class` (read only)	`reference`
`access` (read only)	• `public` • `protected` • `private`
`virtual` (read only)	`boolean`

Class Class

Table B.51 lists the Class AppleScript class properties. The plural reference to use would be Classes. The elements of this class include:

- `base class` by numeric index
- `member function` by numeric index, and by name
- `data member` by numeric index, and by name

Table B.51 Class AppleScript Class Properties

Name	Property Type
`name` (read only)	`string`
`language` (read only)	• `C` • `C++` • `Pascal` • `Object Pascal` • `Java` • `Assembler` • `Unknown`
`declaration file` (read only)	`file specification`
`declaration start offset` (read only)	`integer`
`declaration end offset` (read only)	`integer`
`subclasses` (read only)	`list of class`
`all subclasses` (read only)	`list of class`

Data Member Class

Table B.52 lists the Data Member AppleScript class properties. The plural reference to use would be Data Members.

Table B.52 Data Member AppleScript Class Properties

Name	Property Type
`name` (read only)	`string`
`access` (read only)	• `public` • `private` • `protected`

Name	Property Type
`static` (read only)	`boolean`
`declaration start offset` (read only)	`integer`
`declaration end offset` (read only)	`integer`

Misc Classes

The Miscellaneous classes allow configuration of Version Control Systems, and Extras for the project settings.

- Extras
- Target
- Target File
- Text Document
- Version Control System Setup

Other classes are used for inheritance functions:

- Application
- Build Progress Document
- Catalog Document
- Class Browser
- Class Hierarchy
- Editor Document
- File
- File Compare Document
- Message Document
- Project Document
- Project Inspector
- Single Class Browser
- Single Class Hierarchy

- Symbol Browser
- ToolServer Worksheet

Extras

Table B.53 lists the Extras class properties.

Table B.53 Extras Class

Name	Property Type
`Full screen zoom`	`boolean`
`External Reference`	• `Think Reference` • `QuickView`
`Use Script Menu`	`boolean`
`Use Editor Extensions`	`boolean`
`Use External Editor`	`boolean`

Target

Table B.54 lists the properties for the Target class. The plural form of Target is Targets. This class inherits all properties and elements of the given class.

Table B.54 Target Class

Name	Property Type
`name`	`string`
`index` (read only)	`integer`
`project document` (read only)	`project document`

Target File

Table B.55 lists the properties for the Target File class. The plural form of Target File is Target Files.

Table B.55 Target File Class

Name	Property Type
`id` (read only)	`integer`
`type` (read only)	• `library file` • `project file` • `resource file` • `text file` • `unknown file`
`index` (read only)	`integer`
`location` (read only)	`file specification`
`path` (read only)	`string`
`linked` (read only)	`boolean`
`link index` (read only)	`integer`
`modified date` (read only)	`date`
`compiled date` (read only)	`date`
`code size` (read only)	`integer`
`data size` (read only)	`integer`
`debug`	`boolean`
`weak link` (read only)	`boolean`
`init before`	`boolean`
`prerequisites` (read only)	`list of list`
`dependents` (read only)	`list`

Text Document

Table B.56 lists the properties for the Text Document class. The plural form of Text Document is Text Documents.

Table B.56 Text Document Class

Name	Property Type
`inherits` (read only)	`document`
`modified` (read only)	`boolean`
`selection`	`selection-object`

Version Control System Setup

Table B.57 lists the VCS Setup class properties.

Table B.57 Version Control System Class

Name	Property Type
`VCS Active`	`boolean`
`Connection Method`	`string`
`Username`	`string`
`Password`	`string`
`Auto Connect`	`boolean`
`Store Password`	`boolean`
`Always Prompt`	`boolean`
`Mount Volume`	`boolean`
`Database Path`	`path information`
`Local Root`	`path information`

Application

The Application class elements are:

- `document` by numeric index, by name, and as a range of elements
- `window` by numeric index, by name, and as a range of elements

Build Progress Document

The plural form of Build Progress Document is Build Progress Documents. This class inherits all properties and elements of the given class.

Catalog Document

The plural form of Catalog Document is Catalog Documents. This class inherits all properties and elements of the given class.

Class Browser

The plural form of Class Browser is Class Browsers. This class inherits all properties and elements of the given class.

Class Hierarchy

The plural form of Class Hierarchy is Class Hierarchies. This class inherits all properties and elements of the given class.

Editor Document

The plural form of Editor Document is Editor Documents. This class inherits all properties and elements of the given class.

File

The File Class plural to use in AppleScripts is Files.

File Compare Document

The plural form of File Compare Document is File Compare Documents. This class inherits all properties and elements of the given class.

Message Document

The plural form of Message Document is Message Documents. This class inherits all properties and elements of the given class.

Project Document

The plural form of Project Document is Project Documents. This class inherits all properties and elements of the given class.

Project Inspector

The plural form of Project Inspector is Project Inspectors. This class inherits all properties and elements of the given class.

Single Class Browser

The plural form of Single Class Browser is Single Class Browsers. This class inherits all properties and elements of the given class.

Single Class Hierarchy

The plural form of Single Class Hierarchy is Single Class Hierarchies. This class inherits all properties and elements of the given class.

Symbol Browser

The plural form of Symbol Browser is Symbol Browsers. This class inherits all properties and elements of the given class.

ToolServer Worksheet

The plural form of ToolServer Worksheet is ToolServer Worksheets. This class inherits all properties and elements of the given class.

Coding with CodeWarrior IDE and Apple Events

You may want to use low-level Apple Events instead of writing AppleScripts if you are producing tools or programs that need to control the CodeWarrior IDE while they are running. Third-party editors or browsers, and other tools, might require this capability.

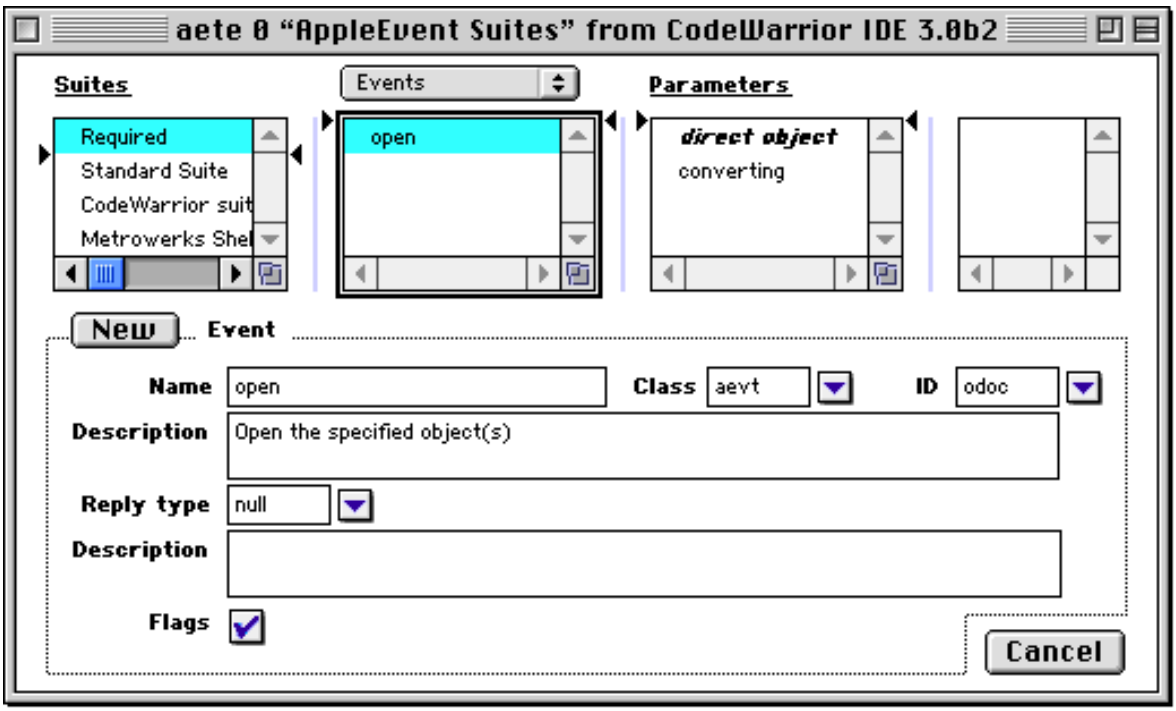
For documentation on using low-level Apple Events in your program code, refer to *Inside Macintosh: Interapplication Communication*

(Addison-Wesley) for a discussion of how to use the Apple Events portion of the Mac OS Toolbox.

There is some example code available that shows how to send Apple Events to the CodeWarrior IDE. You can find it on the CodeWarrior Reference CD in the `CodeWarrior Examples` folder, under the `MacOS Examples` folder. This code is a starter project for your work, and you will need to verify the code for proper operation. It is not intended to be a commercially-shipping product.

Largely, you will need to inspect the CodeWarrior IDE's '`aete`' and '`aedt`' resources using a resource editor to see what the low-level codes are to control the IDE. Figure B.1 shows an example view of what this might look like using the Resorcerer 2.0 resource editor. Rather than document all the low-level codes required to control the IDE, using a resource editor is the best solution to learn the low-level codes for now. With new innovations for the IDE on the horizon, the low-level codes may be documented at a later date.

Figure B.1 Resorcerer 2.0 View of the CodeWarrior IDE 'aete' resource



C

Mac OS CodeWarrior and MPW ToolServer

This chapter describes using the MPW ToolServer development environment from within the CodeWarrior IDE.

Using MPW ToolServer Overview

The topics in this chapter are:

- About ToolServer
- Installing ToolServer
- Starting ToolServer
- Executing MPW Commands
- Looking Up MPW 411 Documentation
- Creating MPW Tool Projects

About ToolServer

ToolServer is a stand-alone version of Apple's MPW (Macintosh Programmer's Workshop) development environment. You can use ToolServer to execute MPW commands, MPW tools, and MPW scripts from the ToolServer Worksheet in CodeWarrior. You can also use it to look up information in MPW's 411 on-line reference database from any CodeWarrior Editor window. This database contains information from *Inside Macintosh,* DTS (Developer Technical Support) Technical Notes, or the *MPW Command Reference.*

Table C.1 describes some of the more popular MPW tools.

Table C.1 Useful ToolServer tools

This command	Does this
`Directory` *folder*	Without an argument, displays the current folder. With an argument, sets the current folder to the argument.
`Files`	Lists the files in the current folder. Use the wildcard ≈ to limit the search (To type ≈, press Option-x). For example, the command `Files w≈` lists all files that begin with the letter w.
`Duplicate` *files target*	Makes duplicates of *files* in the folder *target*.
`Backup -from` *f1* `-to` *f2*	Backs up the files in folder *f1* to the folder *f2*. You can specify which files to back up.
`Rez` *file*	Creates a resource file by compiling the textual description in *file*.
`Derez` *file*	Creates a textual description of resources by disassembling the resource fork of *file*.
`Compare` *file1 file2*	Compares two versions of a file and displays any differing lines.

This command	Does this
`File` *file*	Opens *file* in a CodeWarrior Editor window. For example, the command `File Test.c` opens the file Test.c in the current folder.
`Line` *number*	Selects the line numbered *number* in the front-most CodeWarrior Editor. You'll usually use `Line` in a script with `File`.
`Search` *files pattern*	Displays a list of all lines within *files* that contain *pattern*. For example, `Search /≈Th≈/ ≈.c` searches all files ending in `.c` for words containing the characters Th (To type ≈, press Option-x).

ToolServer and MPW Shell Differences

Unlike the MPW Shell, MPW ToolServer does not let you execute commands that add menus or control windows. However, you can use the File and Line commands to open a CodeWarrior editor window and find a line in it.

Installing ToolServer

This section explains how to install the ToolServer application on your computer's hard drive. To complete this installation, you will need to copy some of the files from the MPW folder on the CodeWarrior CD.

1. **Copy the ToolServer 3.4.1.sit Stuffit archive file from the CodeWarrior Reference CD onto your hard drive.**

   This archive is located in the `Apple Development Tools` folder.

2. **Install the Stuffit Expander application (from the CodeWarrior Tools CD) to your hard drive.**

   You will need to use the Stuffit Expander to unpack the ToolServer archive. Just drag the `ToolServer 3.4.1.sit` file onto Stuffit Expander.

   You can put the ToolServer folder anywhere on your hard drive. CodeWarrior will find it no matter where you place it.

3. **Make sure that you only have one copy of the ToolServer application on your system.**

4. **If you are using a PowerPC Mac OS computer, make sure that the StdCLibInit extension is installed.**

   This extension is located in the `Extensions` folder of your `System Folder`. You can drag it to the `Extensions` folder from the CodeWarrior Tools CD, found in the `System Folder Items` folder.

5. **Create an alias to the ToolServer Tools folder.**

   The Tools folder is located inside the folder that you just installed, and is the same folder that contains the ToolServer application. Then move this alias to the `Metrowerks CodeWarrior` folder, the same folder that the CodeWarrior IDE is in.

6. **Copy any MPW tools you'll need into the ToolServer Tools folder that you just installed.**

   If you installed the MPW Shell onto your hard drive, you can make aliases of the tools in the MPW `Tools` folder and place them into the ToolServer `Tools` folder, which is in the same folder as the CodeWarrior IDE.You can copy any tool you want, but you must copy at least Commando, GetFileName, and GetListItem. If you want to use the MPW 411 database, copy the `Get` tool also. The 411 database is useful for looking up Macintosh Toolbox calls and other information. If you'll be using the `Rez` or `Derez` resource tools from MPW, copy over these tools too.

7. **Create a folder named Interfaces.**

   Create a folder named `Interfaces` in the same folder that contains the ToolServer application you installed.

8. **If you do not want to use the MPW 411 on-line database, delete the MWStartup•411 script.**

   The `MWStartup•411` file is in your ToolServer folder. It asks you to locate the MPW 411 database.

9. **Copy the MPW folder named RIncludes into the new Interfaces folder.**

   This folder is located inside the `Interfaces` folder in the MPW folder on the CodeWarrior CD.

   ToolServer is now ready for use with CodeWarrior.

Starting ToolServer

To start ToolServer, select Start ToolServer from the Tools Menu (Mac OS). CodeWarrior starts up ToolServer, creates a ToolServer Worksheet window and adds the ToolServer menu to the menubar. The title of the ToolServer menu is the icon .

When ToolServer starts up, it executes these scripts, in the order listed in Table C.2.

Table C.2 ToolServer initialization scripts

This script…	Sets up…
StartupTS	The variables for ToolServer and user scripts.
UserStartupTS, and all other scripts whose names start with *UserStartup•*	Your own ToolServer customizations.

This script…	Sets up…
MWStartup	A link between CodeWarrior and ToolServer so File and Line commands work properly.
MWStartup•411	A link between CodeWarrior and MPW's 411 database, so the Lookup Symbol and Insert Template commands in the ToolServer menu work correctly. Delete this file if you don't use 411.
All other scripts whose names start with *MWStartup•*	Your customizations that rely on CodeWarrior being linked to ToolServer.

Finding the 411 database

If you didn't delete the MWStartup•411 script, ToolServer will prompt you to help it find the 411 database.

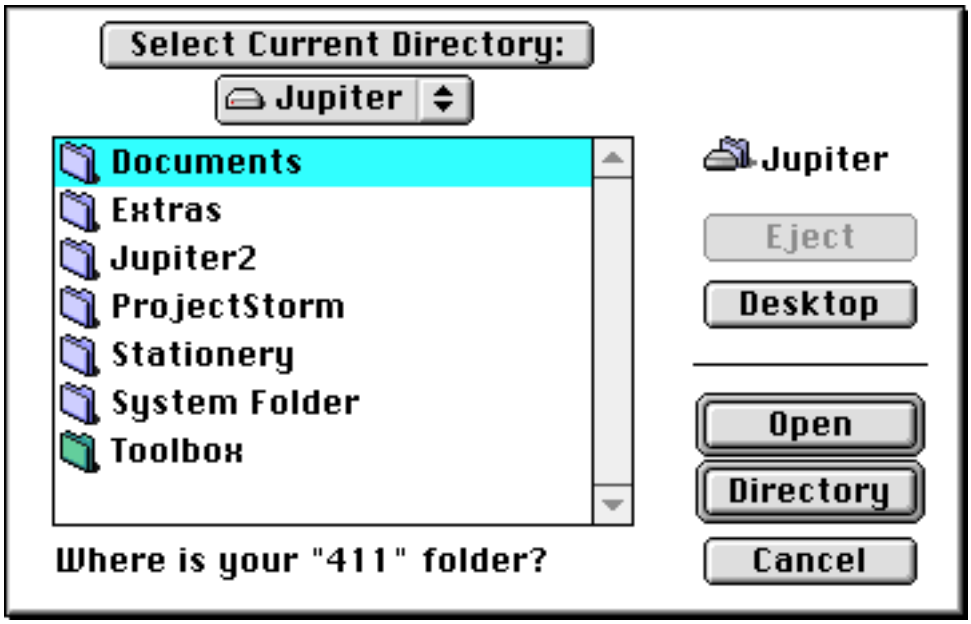
When ToolServer beeps, choose ToolServer from the application menu.

ToolServer displays a dialog, shown in Figure C.1, asking you to find the 411 database. If don't want to use 411, click Cancel.

The dialog disappears. ToolServer continues executing the rest of your startup scripts, and CodeWarrior disables the commands that use 411.

TIP: Delete the MWStartup•411 file as soon as possible if you don't plan to use 411 with ToolServer, so that ToolServer won't display this dialog the next time you start it.

Figure C.1 Finding the 411 Folder



If you want to use 411, select the folder that contains the database and press the Directory button. The dialog disappears. ToolServer saves the location of the database to a file and continues executing the rest of your startup scripts. CodeWarrior enables the commands that use 411.

The next time MWStartup•411 executes, ToolServer will read the location from the file and won't ask you to find it again.

Choose CodeWarrior from the application menu (in the upper right-hand corner of your Macintosh screen).

CodeWarrior comes to the front, with the ToolServer Worksheet at the front.

Executing MPW Commands

CodeWarrior provides several ways to execute MPW commands, tools, and scripts:

- Using the ToolServer WorkSheet

- Using the ToolServer Tools Menu
- Using a Commando Dialog Box

Using the ToolServer WorkSheet

The ToolServer Worksheet is a special window. It lets you enter commands, which CodeWarrior sends to ToolServer. CodeWarrior then writes the results of the command back into the window.

To edit text in the worksheet, use the same commands that you use in any Editor window.

Executing a command

To execute a command, script, or tool that you have installed in ToolServer from the worksheet, type the command and press the Enter key. If you don't have an Enter key on your keyboard, you can use Command-Return instead.

CodeWarrior sends the command to ToolServer and prints ToolServer's response directly below the command's line.

If you press Return, instead of Enter or Command-Return, CodeWarrior just moves the cursor to the next line and does not execute your command.

Re-executing a command, script, or tool

To re-execute a command, script or tool, place the cursor somewhere on the command's line and press the Enter key or Command-Return.

CodeWarrior sends the command to ToolServer and prints ToolServer's response directly below the command's line. The new response appears above any old response.

You can also edit a command in the worksheet window before you re-execute it.

Using the ToolServer Tools Menu

An example of the ToolServer Tools Menu (Mac OS), shown in Figure C.2, is a submenu under the ToolServer menu. This section describes how to add commands to that menu and execute them.

Figure C.2 A ToolServer Tools submenu

Executing a script that's in a CodeWarrior Editor window

Bring the script's Editor window to the front, then choose **Execute as ToolServer Script** from the ToolServer menu, as shown in Figure C.3.

Figure C.3 ToolServer Menu

Adding a command, script, or tool to the ToolServerTools submenu

To add a command, tool, or script to the ToolServer Tools submenu, make an alias to the command, tool, or script on your hard disk.

Although you can use the actual command or script, using an alias helps save disk space.

Move the alias into the `(ToolServer Tools)` folder inside your `Metrowerks CodeWarrior` folder.The `(ToolServer Tools)` folder is in the folder that contains the CodeWarrior IDE, and not in the folder that contains ToolServer.

The next time you look at the ToolServer Tools submenu of the ToolServer menu, the command or script will appear.

Executing a command, script, or tool from the ToolServer Tools menu

To execute a command, tool, or script from the ToolServer Tools submenu, first choose the command or script from the ToolServer Tools submenu of the ToolServer menu (Figure C.2).

If the item is a command with a commando dialog box, ToolServer displays the commando dialog box. If the item is a command with no commando dialog box, ToolServer displays a dialog box that lets you enter command-line arguments. If the item is a script, ToolServer executes it immediately.

CodeWarrior displays ToolServer's response in the ToolServer Worksheet.

Using a Commando Dialog Box

ToolServer lets you use Commando dialog boxes, which are dialog boxes that help you compose a syntactically perfect command, even when you don't know the command's syntax.

This section describes how to bring up these dialogs with the Commando command in the ToolServer menu. You can also bring up a Commando dialog box by typing an ellipsis (Option-;) after a command, like this:

```
rez …
```

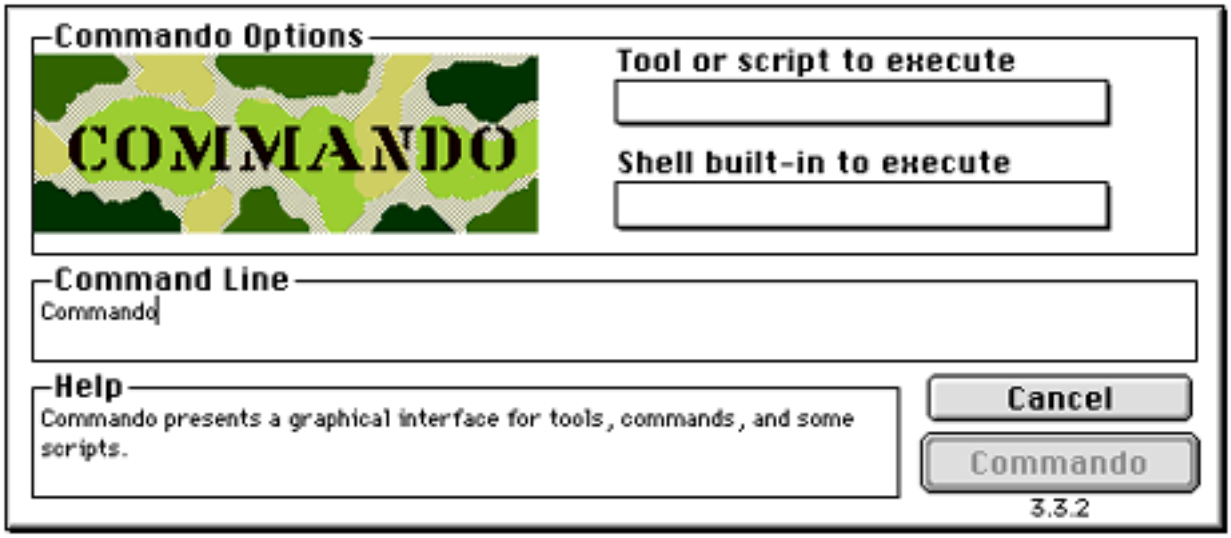
Note that not all commands have Commando dialogs. In order for a command to emit a Commando dialog box for you to use, the developer of the command had to do special things to make the tool Commando-aware.

Executing a command, script, or tool with Commando

Choose Commando from the ToolServer menu or enter the `Commando` command in the ToolServer worksheet (press Enter or Command-Return to execute your command).

ToolServer comes to the front and displays the Commando dialog box, as shown in Figure C.4.

Figure C.4 The Commando dialog

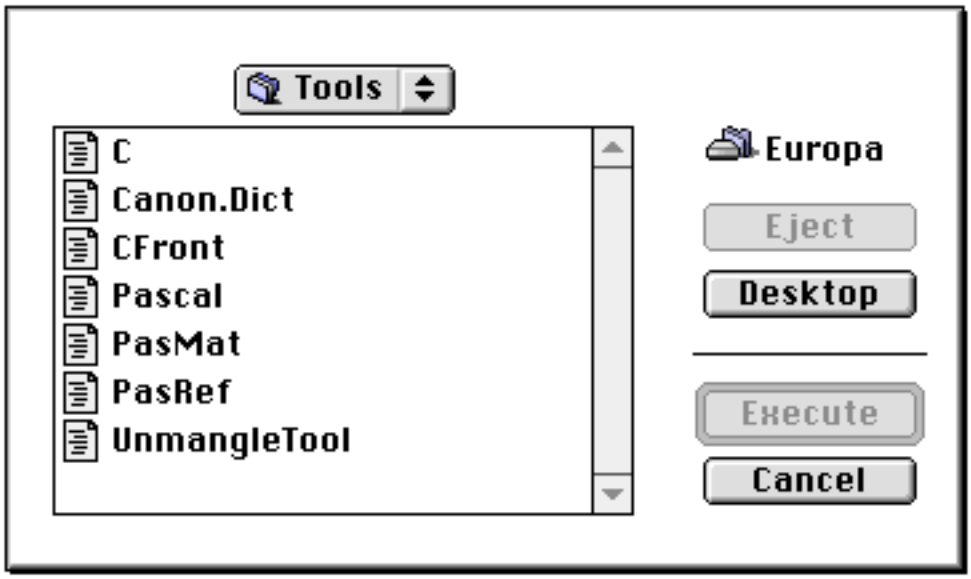


Next, choose the command or script to execute. You can execute a built-in command, a tool, or a script.

To execute a built-in command, choose it from the pop-up menu named Shell Built-in to Execute.

To execute a tool or script, choose Select a tool or script to execute… from the pop-up menu named “Tool or script to execute”. ToolServer displays an open file dialog, as shown in Figure C.5. Select the tool or script you want and click OK.

Figure C.5 Finding a command

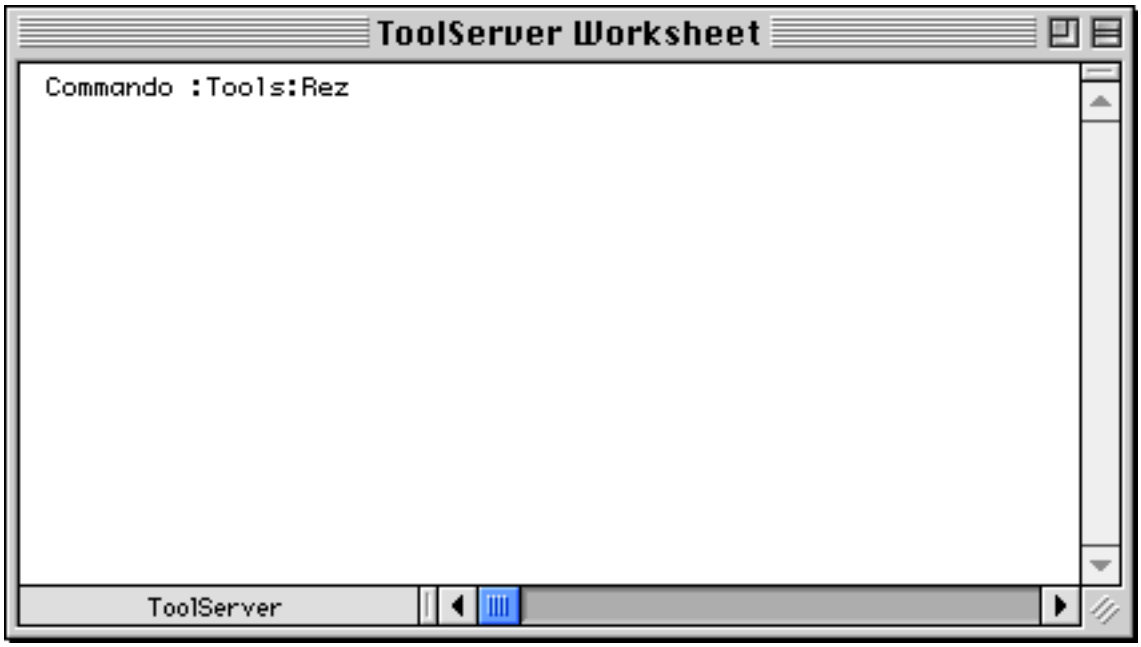


When you have finished configuring the Commando dialog box, click the Commando button in the dialog.

CodeWarrior writes a Commando command-line in the ToolServer WorkSheet.

Then, go back to CodeWarrior by selecting CodeWarrior from the application menu, or clicking on a CodeWarrior window to bring the IDE application to the front. The ToolServer Worksheet may appear as shown in Figure C.6.

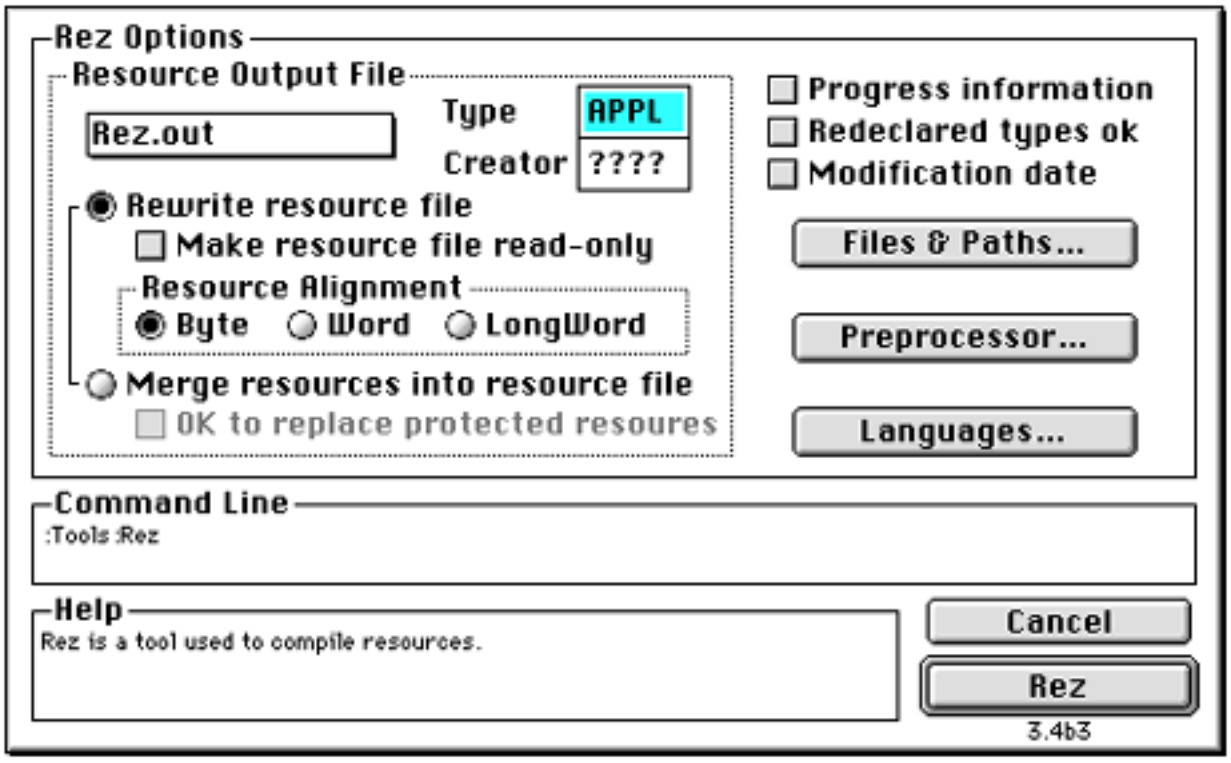
Figure C.6 A Commando command-line in a ToolServer Worksheet



Place the cursor in the Commando command-line and press Enter or Command-Return.

After you press Enter, ToolServer comes to the front and displays the Commando dialog for the command, tool, or script. For example, the dialog for Rez is shown in Figure C.7.

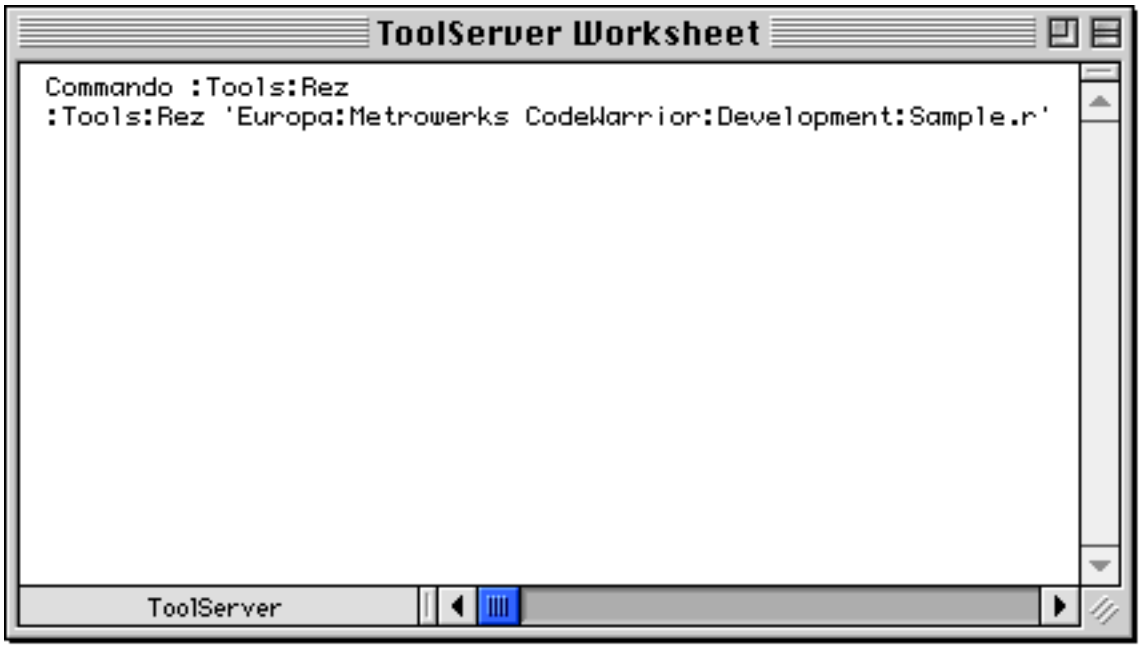
Figure C.7 The Rez Commando dialog box



Fill in the dialog items to correspond to the options you want, and press the button that contains the command's name (the Rez button in this case).

After you press the button, CodeWarrior writes a command line in the ToolServer Worksheet. It will look something like Figure C.8.

Figure C.8 The Rez command-line in a ToolServer Worksheet



Place the cursor in the command-line and press Enter or Command-Return.

ToolServer executes the command and CodeWarrior writes the result into the ToolServer Worksheet.

Looking Up MPW 411 Documentation

CodeWarrior lets you use MPW's 411 database, which allows you to look up information from *Inside Macintosh*, DTS Technical Notes, or *MPW Command Reference*. Before you use it, make sure the MWStartup•411 script is in your ToolServer folder when you start up ToolServer.

Looking up documentation for a symbol

In a CodeWarrior Editor window, select the name of the symbol.

You can select the name of almost anything that might be in *Inside Macintosh*, including functions, types, and variables.

Next, choose **Lookup Symbol** from the ToolServer menu.

If CodeWarrior finds the symbol in the 411 database, it displays the information in a CodeWarrior Editor window named Help. The cursor is at the beginning of the information. You may need to scroll to see all of it.

If CodeWarrior can't find the symbol in the 411 database, it writes an error message in the ToolServer Worksheet and displays information on the last symbol you looked up.

The Help window contains all the 411 information you've ever looked up. To remove old information, just delete it. CodeWarrior won't display it the next time you look something up.

When you're finished reading the information, close the Help window.

CodeWarrior cannot look up more 411 information if the Help window is open.

Looking up a template for a function

In MPW 411, a template is a sample function call which contains the names of the function's arguments.

In a CodeWarrior Editor window, select the name of the function.

You can select the name of any function described in *Inside Macintosh*.

Choose **Insert Template** from the Tools Menu (Mac OS).

CodeWarrior inserts the parameter list after the selected text.

Creating MPW Tool Projects

An MPW tool is a program that you can run under Apple's MPW (Macintosh Programmers Workshop). MPW tools will work with ToolServer, so you can write your own custom tools to enhance the ToolServer environment.

To learn how to write MPW Tools, refer to the *Targeting Mac OS* book on your CodeWarrior Reference CD.

To learn how to install your tool into ToolServer, refer to "Installing ToolServer" on page 515.

To learn how to execute your tool after installing it into ToolServer, refer to "Executing MPW Commands" on page 519.

D

Solaris Utilities

This chapter describes utilities provided in the Solaris version of the CodeWarrior IDE. These utilities are accessible from the Info menu on the right-hand side of the menu bar in the IDE windows. This menu is only available in the Solaris version of the CodeWarrior IDE

Overview

The topics covered in this chapter are:

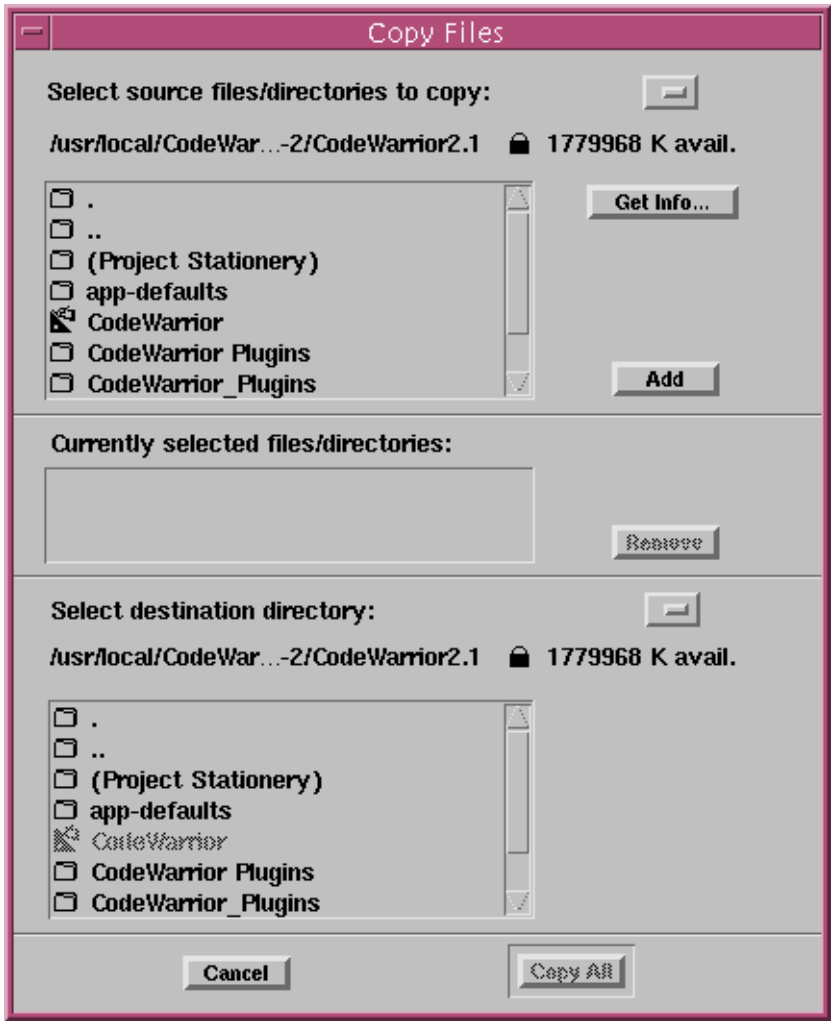
- Copy Files Accessory
- File Systems Facility
- Keyboard Preferences Dialog Box

Copy Files Accessory

When the Copy Files… menu item is chosen, CodeWarrior launches a desk accessory which enables you to copy files and folders between file systems or devices accessible in your Solaris environment from within the CodeWarrior IDE. This includes Macintosh diskettes, AppleDouble, K-AShare, EtherShare, and Partner file systems, and NFS-mounted UNIX files systems. For example, you can use Copy Files to copy a file from a Macintosh diskette into a UNIX directory.

To display the Copy Files dialog box (Figure D.1), choose Copy Files... from the Info menu.

Figure D.1 The Copy Files dialog box



This dialog box contains three sections: Source File/Directory Selection, Currently Selected Files/Directories, Destination Directory Selection

Source File/Directory Selection

This area of the dialog box is used for selecting files to copy from the source volume. It contains the following items:

Path Display

The path display shows the path to the current directory being listed.

Free Space Display

This display shows how much disk space is available on the currently selected volume.

Source File/Directory List

This is a scrollable list of files in the current directory. Double-clicking a directory or volume in the list displays the contents of that directory or volume. Double-clicking a file adds it to the list of files to be copied. Only one item can be selected at a time.

Get Info Button

Pressing this button displays a dialog box that provides file information about the currently selected file or directory in the File/Directory List. If no file or directory is currently selected in the list, an alert message appears with instructions.

Add button

This buttons allows you to add the directory, volume, or file currently selected in the source directory to the list of files to be copied.

Currently Selected Files/Directories

This area of the dialog box is for viewing the files you have selected for copying. It contains the following items:

Selected Files/Directories List

This list displays the truncated path(s) of the file(s) to be copied. As entries are being copied, they are highlighted. After a file is copied, it is removed from the list. If an error occurs during the copying process, the copy is aborted and the list displays the names of files that have not yet been copied.

Remove button

Pressing this button removes the selected files/directories from the Source Files/Directories List.

Destination Directory Selection

This area of the dialog box is for setting the destination directory where files will be placed. It contains these items:

Destination Path Display

This display shows the path to the currently selected destination.

Free Space Display

This display shows how much disk space is available on the destination volume.

Destination File/Directory List

This is a scrollable list of files in the current directory. Double-clicking a directory or volume in the list displays the contents of that directory or volume.

Cancel Button

Pressing this button closes the dialog box without performing the copy operation.

Copy All Button

Pressing this button starts the copy procedure. You are first prompted by a notice to verify the destination directory. Once you verify this, the copy proceeds.

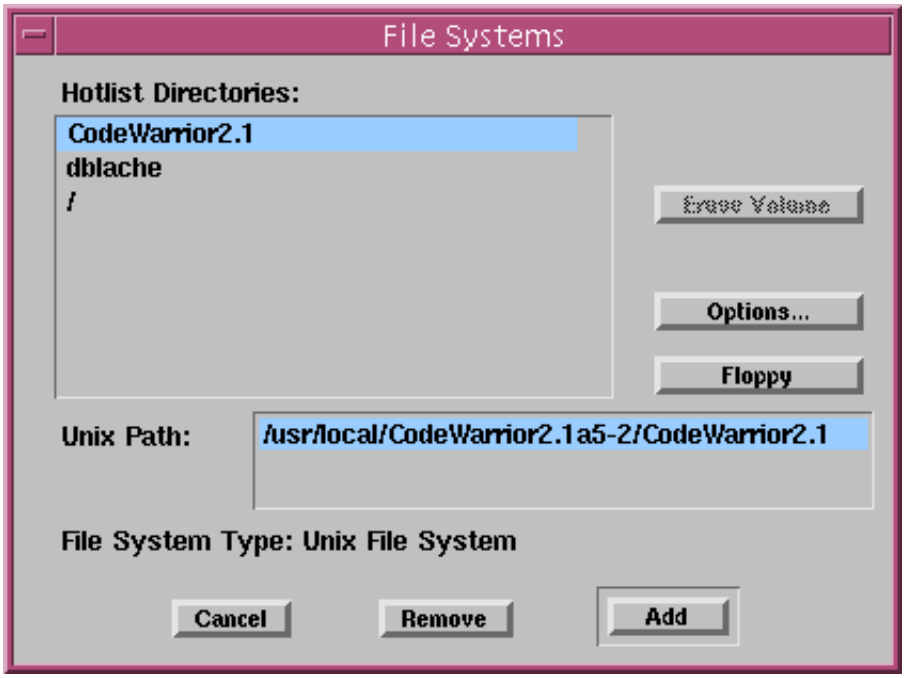
File Systems Facility

CodeWarrior for Solaris provides the File Systems Facility, which enables you to mount volumes on your workstation. When the File Systems… menu item is chosen from the Info menu, the File Systems Facility dialog box is displayed (Figure D.2).

The File Systems Dialog Box

The File Systems dialog box contains the following items: Hotlist Directories, Erase Volume Button, Options Button, Floppy Button, Unix Path Text Box, File System Type Display, Cancel Button, Remove Button, Add Button.

Figure D.2 The File Systems dialog box



Hotlist Directories

This list shows all of the volumes currently mounted on your system. You can select any volume listed and then unmount or erase it,

or display the volume's full pathname and file system type, according to the options you choose below.

Erase Volume Button

Pressing this button deletes all files in the chosen volume.

Options Button

When this button is pressed, the Options dialog box is displayed. This dialog box lets you set line breaks in files you mount. See "The File System Options Dialog Box" on page 537 for more information.

Floppy Button

Lists any diskette in the diskette drive (in the Unix Path field) and enables the Mount button.

Unix Path Text Box

This text box contains the full UNIX pathname of the chosen volume. If no volume is currently chosen, you can type a path into this area to mount a volume by its path.

File System Type Display

This display shows the type of format for the chosen volume--UNIX or HFS.

Cancel Button

Pressing this button closes the dialog box without mounting any file systems.

Remove/Eject Button

If the chosen volume is on a hard drive, this option unmounts it. If the chosen volume represents a diskette, this option ejects the diskette from the diskette drive.

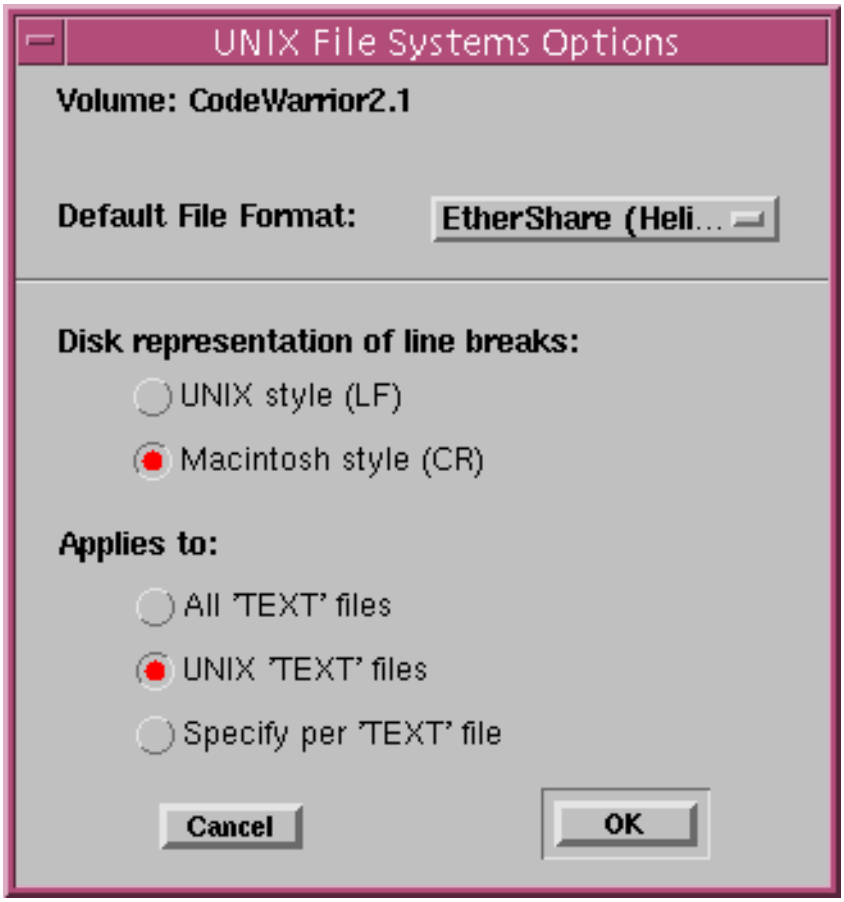
Add Button

Pressing this button mounts the currently selected volume. and closes the dialog box.

The File System Options Dialog Box

Some applications may have difficulty representing line breaks appropriately as their files move from foreign operating systems to UNIX systems. The File System Options dialog box enables you to solve this problem by explicitly choosing how to represent line breaks in certain types of files. To view this dialog box, press the Options button described above.

Figure D.3 The File System Options dialog box



This dialog box contains the following items: Volume Display, Default File Format, "Disk representation of line breaks" button group, "Applies to" button group, Cancel button, OK button.

Volume Display

This display shows the name of the volume which will be affected by the settings in this dialog box.

Default File Format

This popup menu lets you specify the file format for any new files that will be created on the volume in question.

Disk representation of line breaks

This group of buttons determines how the IDE will convert line break characters when it reads or writes files on the current volume.

UNIX Style (LF)

If this option is chosen, line breaks are replaced with line feeds whenever a file on the current volume is read or written by the CodeWarrior IDE.

Macintosh Style (CR)

If this option is chosen, line breaks will be replaced with carriage returns whenever a file on the current volume is read or written by the CodeWarrior IDE.

Applies to

This group of buttons determines which files will be affected by your settings in the File Systems dialog. Only one of these buttons may be selected at a time.

All 'TEXT' Files

If this option is selected, conversion will be applied to all text files read or written by the IDE on the current volume.

UNIX 'TEXT' Files

If this option is selected, conversion will be applied only to UNIX (non-Macintosh) text files read or written by the IDE on the current volume.

Specify per 'TEXT' Files

If this option is selected, a dialog box will be displayed each time a text file is read or written by the IDE, allowing you to specify how line breaks should be converted for that file.

Cancel button

Pressing this button dismisses the dialog box without applying any settings.

OK button

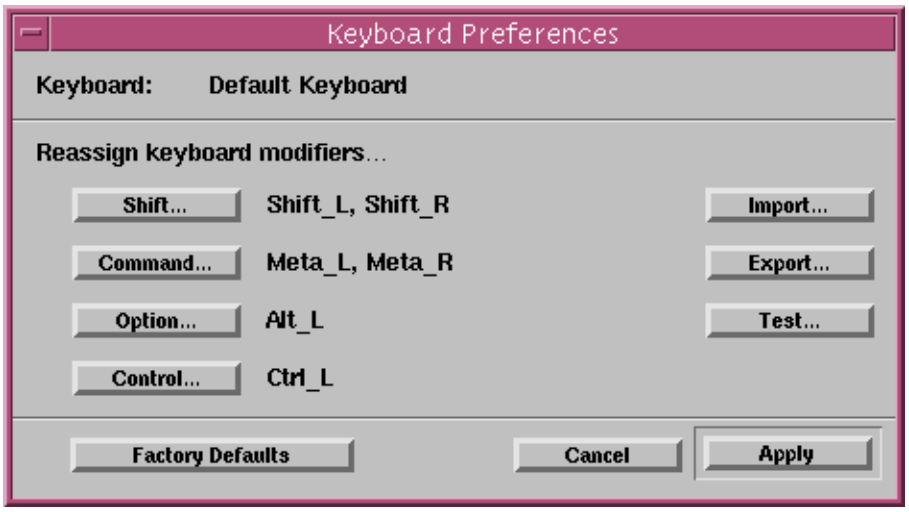
Pressing this button applies the selected settings and dismisses the dialog box.

Keyboard Preferences Dialog Box

Modifier keys are used in combination with other keys to generate keyboard shortcuts to certain IDE functions. For instance, to execute the Ctrl/Command - N keyboard shortcut, you depress the Ctrl or Command modifier key, and while holding it down, press the N key on your keyboard.

The Keyboard Preferences dialog box (Figure D.4) allows you to define which physical keys on your keyboard correspond to certain modifier keys used in the CodeWarrior IDE. To display the dialog box, choose **Keyboard Preferences** from the Info menu.

Figure D.4 The Keyboard Preferences dialog box



This dialog box contains the following items: Keyboard Display, Reassign Keyboard Modifiers Section, Import Button, Export Button, Test Button, Factory Defaults Button, Cancel Button, Apply Button.

Keyboard Display

This display shows the active keyboard.

Reassign Keyboard Modifiers Section

The Shift, Command, Option, and Control buttons in this section correspond to modifier keys used in the CodeWarrior IDE.

Pressing one of these buttons will cause CodeWarrior to scan the keyboard and record keys that are pressed until a key is released. The key(s) that are pressed during the scan will be mapped to the corresponding modifier key. Once scanning is complete, a list of keys that were pressed will appear to the right of the button.

Once the Apply button is pressed, the changes you have made go into effect. From this point on, when you type any of these keys, CodeWarrior will respond as if the corresponding modifier key has been pressed.

For example, to map both the left and right Shift keys on your keyboard to the Shift modifier key, follow these steps:

1. Press the Shift button in this dialog box.
2. Press and hold down both of the Shift keys on your keyboard.
3. Release the Shift keys on your keyboard.

Shift_L and Shift_R should now appear next to the Shift button in the dialog box.

Import Button

Press this button to display a file open dialog, which will allow you to select a file to import keyboard modifier settings from.

Export Button

Press this button to display a file save dialog which will allow you to export the current keyboard modifier settings to a file on disk.

Test Button

Press this button to test your settings.

Factory Defaults Button

Press this button to reset the settings to the factory defaults.

Cancel Button

Press this button to dismiss the dialog box without applying your changes.

Apply Button

Press this button to apply any changes you have made and dismiss the dialog box.

Index

Symbols

(ToolServer Tools) folder 522
`.dump` 337
"Weak" link 351
'ckid' resources 361

Numerics

80x86 16

A

About Metrowerks command 407, 410
Absolute Path 302
access paths 298
    recursive search 298
Add Default button 301
Add File command 389
Add Files command 389
Add Window command 78, 389
Advanced VCS operations 366
Alert Yourself After Build option 328
All Exceptions command 408
Alpha 85
Always Search User Paths 300
Always Show Login Dialog option 358
Anchor Floating Toolbar command 410
Apple Event 22
Apple Menu 410
    About Metrowerks command 410
AppleScript 22
arrow keys 43
Attempt to use dynamic type of C++, Object Pascal and SOM objects 272
Automatic updating 331
Automatically launch applications when SYM file opened 276

B

Background Color 262
backup files 66, 111
`Backup` MPW command 514
Balance command 147, 381
Balance While Typing option 263
Balancing punctuation 147
Batch search 180
BBEdit 85
BBEdit Extensions Menu 414
BeOS 16
Break on C++ Exception command 398
Break on Java Exceptions command 398
Breakpoints Window command 404
Bring Up To Date command 328, 391
browser 201–233
    activating 202, 306
    analyzing inheritance 231
    base classes in hierarchy 221
    base classes in single-class 218
    catalog view 204, 209
    classes pane 214
    customizing windows 233
    data members pane 215
    declaration button 219
    editing code 231
    file button 216
    finding function overrides 231
    hierarchy view 206
    identifier icon 216
    including subprojects 306
    interface 207–224
    lines in hierarchy 220
    list button 214
    member functions pane 214
    multi-class 211–216
    multi-class hierarchy 219–221
    navigating code with 208
    opening source file 229
    orientation button 216
    resize bar 213
    saving windows 233
    seeing declaration 230
    seeing function definition 230
    seeing MFC in 232
    seeing PowerPlant in 232
    show hierarchy button 219
    showing data in single-class 219
    showing subclasses in hierarchy 220
    single-class 217
    single-class hierarchy 221
    source pane 215
    strategy 202–207

symbol window 222
synchronized class selection 214, 217, 220
using 225–233
view 204
viewing options 202
Browser Catalog Window command 402
Browser Display preference panel 260
Build Progress Window command 403
Build Settings 239
build target 25
build, alert when completed 328

C

C 17
C String command 400
C++ 17
Cache Edited Files Between Debug Sessions 276
catalog view 204
catalog window
browser 209
Catch 68K Debugger and DebugStr Traps 278
Catch PowerPC Debugger and DebugStr Traps 278
category pop-up menu, in category window 210
CFM68K 351
Change button 305
Changing syntax highlighting colors 268
Character command 400
Check Syntax command 390
Checkin Options 370
Checkout Options 370
Checkout Status Column 48
ckid resources 361
Class for Debugging 278
Class Hierarchy Window command 403
classes pane in browser 214
Clear All Breakpoints command 397
Clear All Watchpoints command 398
Clear Breakpoint command 396
Clear command 147, 380
Clear Floating Toolbar command 410
Clear Watchpoint command 397
Clear Window Toolbar command 409
Close All command 376
Close command 376
Close non-debugging windows 273
code navigation pop-up menu 208
CodeWarrior IDE 22
CodeWarrior IDE command 407
Collapse Window command 402
Color Syntax option
and printing 118
Commando command 411, 523
comments, coloring 267
Common Sub-Expressions 314
Compare Files command 388
`Compare` MPW command 514
Compile column 83
Compile command 391
Compiler 311
Compiler Relative Path 303
Compiler Thread Stack (K) 239
Compiling 317–349
one file 321
project 319
selected files 321
compiling
source files 321
Compiling and Linking
choosing a compiler 318
compiling files 320–321
debugging 325
disassembling 336
guided tour 337–340
link map 326
making a project 323
options 328
overview 317
plugin compilers 318
precompiling headers 329–336
preprocessing 336
removing binaries 327, 328
removing object code 327
running 324
setting file extension 319
speeding 328
Synchronizing Modification Dates 326
Confirm “Kill Process” when closing or quitting 276

Confirm invalid file modification dates when debugging 276
Connect on Open option 358
Contents command 407
Context Popup Delay option 265
context pop-up menu 208
Controlling syntax highlighting within a window 268
conventions 18
  figures 19
  host terminology 19
  keyboard shortcuts 20
Copy command 147, 380
Copy Files… 416
Copy Propagation 315
Copy to Expression command 399
Create Folder 53
Create New Group command 389
Create New Segment command 390
Create New Target command 389
custom keywords, coloring 268
customizing toolbars 280–288
Cut command 147, 380

D

data members pane, in browser 215
Data Menu 399–401
  C String command 400
  Character command 400
  Copy to Expression command 399
  Default command 400
  Enumeration command 400
  Fixed command 401
  Floating Point command 400
  Fract command 401
  Hexadecimal command 400
  New Expression command 399
  Pascal String command 400
  Show Types command 399
  Signed Decimal command 400
  Unsigned Decimal command 400
  View Array command 399
  View As command 399
  View Variable command 399
date caching 328
Dead Code Elimination 314
Dead Store Elimination 315
dead stripping 45
Debug all Java class files in directory hierarchy 276
Debug Column 97
Debug Info Marker 97
Debug Menu 395–398
  Break on C++ Exception command 398
  Break on Java Exceptions command 398
  Clear All Watchpoints command 398
  Clear Breakpoint command 396
  Clear Watchpoint command 397
  Disable Breakpoint command 397
  Disable Watchpoint command 398
  Enable Breakpoint command 397
  Enable Watchpoint command 398
  Hard Reset command 396
  Hide Breakpoints command 397
  Kill command 395
  Reset command 395
  Set Breakpoint command 396
  Set Watchpoint command 397
  Show Breakpoints command 397
  Step Into command 396
  Step Out command 396
  Step Over command 396
  Stop command 396
  Switch To Monitor command 398
  View Memory As command 400
  View Memory command 399
Debugger command 407
Debugger, Display panel 271–272
Debugger, Java Settings panel 278
Debugger, MetroNub panel 277–278
Debugger, Settings panel 275–277
Debugger, Windowing panel 272–275
Debugger, x86 Settings panel 279
debugging
  configuration 324
  generating debugging information 97
  on multiple monitors 274
  project 37
Default command 400
default file name extensions 309–311
Default size for unbounded arrays 272

Default Text File Format 265
Defining Symbols for C/C++ 333
Defining Symbols for Pascal 334
`Derez` MPW command 514
`Directory` MPW command 514
Disable Breakpoint command 397
Disable Debugger command 394
Disable Watchpoint command 398
Disassemble command 391
Disassembling source code 336
Display panel, Debugger 271–272
Do nothing to non-debugging windows 272
documentation viewers 24
Don't step into runtime support code 277
DOS text files 113
Drag & Drop editing support 264
Drag and Drop text 146–147
Dump internal browse information after compile 307
dump memory 399
`Duplicate` MPW command 514
Dvorak KCHR Support 251
Dynamic Scrolling option 263

E

Edit Menu 379–382
- Balance command 381
- Clear All Breakpoints command 397
- Clear command 380
- Copy command 380
- Cut command 380
- Insert Reference Template command 381
- Multiple Redo 379
- Multiple Undo 379
- Paste command 380
- Preferences command 382
- Redo command 379
- Select All command 381
- Shift Left command 381
- Shift Right command 381
- Target Settings command 382
- Undo command 379
- Version Control Settings command 382

editing
- in browser 231
- editing, redoing 149
- editing, undoing 149

Editor 129–155
- adding text 143
- balancing punctuation 147
- color syntax 150
- configuration 137–141
- deleting text 144
- drag and drop 264
- finding a routine in 151
- font 138
- font preferences 263
- Go Back and Go Forward 154
- go to line number 154
- guided tour 130–137
- markers 151–153
- moving text 146–147
- navigating text 150–155
- opening related file 153
- overview 129, 353
- panes 139–140
- saving window settings 140
- selecting text 144
- text editing 141–150
- text size 138
- third-party editors 85
- third-party support 244
- undoing changes 148–149
- user interface elements 130

Editor Extensions Menu 414
Editor Settings preference panel 261
Embedded PowerPC 16
Enable 398
Enable Automatic Toolbar Help (Mac OS) 246
Enable Breakpoint command 397
Enable Debugger command 394
Enable Debugging command 394
Enable Watchpoint command 398
End Key 43, 143
Enter 'Find' String command 173, 384
Enter 'Replace' String command 384
Entire Word checkbox 177
Enumeration command 400
Error Button 338
Error Messages 341

compiler 342
Errors & Warnings Window command 403
Exceptions in Targeted Classes command 408
Execute as ToolServer Script command 412, 522
Exit command 378
Expand Window command 402
Expressions Window command 404
extension, file name 309
external editor support 244
Extra Information Button 338

F

Factory Settings button 237, 292
Faster Execution Speed 314
fat libraries 351
FDI 244
figure conventions 19
file button, in browser 216
File Control pop-up 48
File Mappings List 309
File Menu 374–379
Close All command 376
Close command 376
Exit command 378
Find and Open 'Filename' command 376
Find and Open File command 376
New command 375
New Empty Project command 375
New Project command 375
Open command 375
Open Recent command 375
Page Setup command 378
Print command 378
Print Setup command 378
Quit command 378
Revert command 378
Save A Copy As command 378
Save All command 377
Save As command 377
Save command 377
Switch to MW Debugger command 377
`File` MPW command 515
file name extensions 309
default settings 309–311
file name suffix 309
File Path Caption 135
File Sets List 168
File Sets Pop-up Menu 168
File Sets, saving 187
File Systems… 416
Files 101–119
closing 115–117
creating 101
opening 153
opening existing 102–108
overview 101
printing 117–119
printing options 117
read-only 362
reverting to saved 119
saving 108–113
saving file sets 187
saving in default text format 265
selecting 70
source code control 359–372
`Files` MPW command 514
Find and Open 'Filename' command 376
Find and Open File command 376
Find command 383
Find Definition & Reference command 156, 387
Find Definition command 156, 386
Find in Next File command 383
Find in Previous File command 384
Find Next command 173, 383
Find Previous command 173, 176, 383
Find Previous Selection command 385
Find Reference command 156, 387
Find Reference Using (Mac OS) 247
Find Selection command 384
Find Window
guided tour 161–172
finding all implementations of function 222
Fixed command 401
Flags 311
Flashing Delay option 264
Floating Document Interface 244
Floating Point command 400
Floating Toolbar 282
Font & Tabs preference panel 265
Font Preferences 263

Fract command 401
functions, finding all 222

G

Get Options 370
Global Register Allocation 314
Global Variables Window command 404
Go Back command 388
Go Forward command 388
Go To Line command 388
`grep` 191–200
groups 79–83
    creating 80
    moving 79
    naming 82
    removing 81
    selecting 70

H

Hard Reset command 396
header files
    opening 108
    precompiling 329–336
Help Menu 406–407
    About Metrowerks command 407
    CodeWarrior IDE command 407
    Contents command 407
    Debugger command 407
    How to Use Help command 407
    Keys command 407
    Mac OS 407
    Windows 407
Hexadecimal command 400
Hide Breakpoints command 397
Hide Floating Toolbar command 409
Hide non-debugging windows 272
Hide Window Toolbar command 409
hierarchy expansion triangle 220
hierarchy view, browser 206
Home Key 43, 143
host terminology conventions 19
How to Use Help command 407

I

IDE 15, 22, 31
    guided tour 35–38
    installation 31
    online documentation 155
    Scripts menu 413
IDE Extras preference panel 242
identifier icon in browser 216
Ignore Case checkbox 177
Ignored by Make option 312
Import Weak command 351
In variable panes, show variable types by default 271
Include File Cache Size (Mac OS) 241
Info Menu 415
Initialize Before command 351
Insert Reference Template command 381
Insert Template command 412
Installation 31
Integrated Development Environment 15
Intel 16
Interface Pop-up Menu 131
Interfaces File Pop-up menu 84
Interpret DOS and Unix paths (Mac OS) 301

J

Java 16, 17, 25
Java Exceptions Submenu 408
    All Exceptions command 408
    Exceptions in Targeted Classes command 408
    No Exceptions command 408
Java Settings panel, Debugger 278
JView Arguments 278

K

key bindings 247–260
keyboard conventions 20
    Solaris 21
Keyboard Preferences… 416
Keys command 407
keywords, coloring 268
Kill command 395

L

Launchable option 311
Lifetime Analysis 315
line button in hierarchy browser 220
`Line` MPW command 515
Line Number Button 136
line number, going to 154
link order, setting 322
Linker option 297
linker, shared libraries 350
Linking 317–349
list button, in browser 214
Lookup Symbol command 412
Loop Unrolling 315
Loop-Invariant Code Motion 315

M

Mac OS 16, 25, 30
    Help Menu 407
Main Text Color 262, 267
Make command 323, 328, 392
manual style 18
MAP 326
Marker Pop-up Menu 133
MDI 244
member functions pane, in browser 214
memory dump 399
memory requirements 30
Merge Into Output command 351
Merge shared libraries option 351
Message List Pane 340
Message Window
    command 403
    correcting compiler errors 343
    error and warning messages 341
    stepping through messages 342
    using 340–349
Method option 357
MetroNub panel, Debugger 277–278
Metrowerks Visual SourceSafe for Macintosh 48
Microsoft Visual SourceSafe 48
Minimize non-debugging windows (Windows) 273
Minimize Window command 402
MIPS 26
Modification dates, synchronizing 84
Monitor For Debugging 274
MPW 116, 362
    function documentation 528
MPW commands 519
multi-class browser 211–216
multi-class hierarchy, browser 219–221
Multi-file searches 168–172
Multiple Document Interface 244
multiple redo 379
multiple Redo command 149
multiple undo 379
multiple Undo command 149
multiple-monitor debugging 274

N

New Class Browser command 403
New command 375
New Empty Project command 375
New Expression command 399
New Project command 51, 59, 375
No Exceptions command 408
notification of completed build 328
Nucleus 16, 26
number, going to line 154

O

Object Master 85
online help and references 155
online reference database 243
Open command 375
Open Recent command 375
Open Recent Menu 243
Open Scripts Folder command 414
opening file with browser 229
Options 235–288, 289–315
    Access Paths 298–305
    advanced compile options 328
    browser display 260
    Build Extras 305–306
    Custom Keywords panel 312

Editor settings 261
fonts and tabs 265
IDE extras 242
key bindings 247–260
overview 235, 289
preferences 238
syntax coloring 266–270
target settings 293–356
Options Pop-up Menu 134
orientation button, in browser 216
OS-9 16
Others Button 172
Output Directory 297

P

Page Down key 43, 143
Page Setup command 378
Page Up key 43, 143
Palm OS 16, 26
Pane Resize Bar 340
Pane Splitter Controls 136
Panes, in editor window 139–140
Pascal 17
Pascal String command 400
Password field 358
Paste command 147, 380
Peephole Optimization 315
PEF container 351
platform target 25
Play Sound After 'Bring Up To Date' & 'Make' (Mac OS) 242
PlayStation 16, 26
Plugin Diagnostics 240
Port ID 279
Post Linker option 297
PowerPC 30, 351
PowerPC Embedded 26
Precompile command 331, 331–??, 391
`precompile_target` 331
precompiled headers
automatic updating of 331
creating 330
Precompiled option 311
precompiling 329–336
Preferences command 382
Preferences, choosing 238
prefix key 249
Quote Key 249
timeout 250
Preprocess command 391
Preprocessing code 336
preprocessor 336
Print command 378
Print Selection Only 118
Print Setup command 378
Print using Syntax Highlighting option 118
printing
window 118
with syntax coloring 118
Processes Window command 404
Program Arguments 278
Project Headers checkbox 172
Project Information Caption 338
Project Inspector 350
Project Inspector command 403
Project Menu 389–395
Add File command 389
Add Files command 389
Add Window command 389
Bring Up To Date command 391
Check Syntax command 390
Compile command 391
Create New Group command 389
Create New Segment command 390
Create New Target command 389
Disable Debugger command 394
Disassemble command 391
Enable Debugger command 394
Enable Debugging command 394
Make command 392
Precompile command 391
Preprocess command 391
Remove Binaries command 392
Remove Object Code & Compact command 392
Remove Object Code command 392
Remove Selected Items command 390
Re-Search for Files command 393
Reset File Paths command 390
Reset Project Entry Paths command 393

Resume command 394
Run command 394
Set Current Target command 395
Set Default Project command 395
Synchronize Modification Dates command 393
Project Relative Path 303
project stationery 56, 57
about 56
creating 57
folder 57
Project Switch List submenu 61
Project Window
guided tour 42–50
navigating 43
Projector 360, 362
Projects 41–100
adding files 71–79
adding preprocessor symbols to 99
building 56, 319
choosing default project 67
choosing stationery 51
closing 67
compiling 319
creating 56–61
creating groups 80
debug enabling 324
debug setup 97–99
debugging 325
expanding and collapsing groups 68
items saved with 66
making 323
managing files in 68–96
modifying 55
moving around 43
moving files and groups 79
naming 53
new 51
opening existing 61–62
opening with Project Switch List 62
removing files and groups 81
renaming groups 82
revision control 48
running 324
saving 65, 65–66
selecting files 70
selecting files and groups 70–71
selecting groups 70
setting link order 322
settings 293–356
stationery folder 57
stationery, about 56
switching between 61
touching and untouching files 83
updating 322
using stationery 50
projects
building 328
compiling 321, 328

Q

QuickStart 24, 31
Quit command 378
Quote Key prefix 249

R

read-only files 362
Recent Documents 243
Recent Projects 243
Recent Strings pop-up menu 175, 178
recompiling 321
recompiling files 321
recursive search of access paths 298
Redo command 379
Registers Windows command 405
regular expressions 191–200
Relaxed C Popup Parsing 264
Remember Password option 358
Remote IP Address 279
Remove a file set command 188
Remove Binaries command 392
Remove button 305
Remove Files command 81
Remove Object Code & Compact command 392
Remove Object Code command 392
Remove Selected Items command 390
Replace & Find Next command 385
Replace & Find Previous command 386
Replace All command 180, 386
Replace command 385

Re-Search for Files command 393
Reset command 395
Reset File Paths command 390
Reset Floating Toolbar command 410
Reset Project Entry Paths command 393
Reset Window Toolbar command 409
resize bar, in browser 213
Resource File option 311
Restore Window command 402
Resume command 394
Revert command 378
Revert Panel button 237, 292
revision control 359–365, 405
    advanced operations 366
revision control systems 48
    options 354
`Rez` MPW command 514
Routine Pop-up Menu 151
routine pop-Up menu 132
Run command 394

S

Save A Copy As command 66, 111, 378
Save All Before Build option 264
Save All command 377
Save As command 110, 377
Save command 377
Save Default Window command 402
Save this File Set command 187
Scripts Menu 413
    Open Scripts Folder command 414
Search Menu 382–388
    Compare Files command 388
    Enter 'Find' String command 384
    Enter 'Replace' String command 384
    Find command 383
    Find Definition 386
    Find Definition & Reference command 387
    Find in Next File command 383
    Find in Previous File command 384
    Find Next command 383
    Find Previous command 383
    Find Previous Selection command 385
    Find Reference command 387
    Find Selection command 384
    Go Back command 388
    Go Forward command 388
    Go To Line command 388
    Replace & Find Next command 385
    Replace & Find Previous command 386
    Replace All command 386
    Replace command 385
Search Menu Find Selection command 384
`Search` MPW command 515
searching 172
    find and replace 181–191
    for selection 173
    multi-file 168–172
    selected text 172–174
Select All command 381
Select stack crawl window when task is stopped 277
selected text search 172
selection
    finding 173
    printing 118
Selection Position 263
Set Breakpoint command 396
Set Current Target command 395
Set Default Project command 395
Set Watchpoint command 397
setting link order 322
Settings panel, Debugger 275–277
shared libraries 351
    linker options 350
Shift Left command 148, 381
Shift Right command 148, 381
shortcut conventions 20
    Solaris 21
Show Breakpoints command 397
Show Catalog Window command 402, 403
Show Floating Toolbar command 409
Show Global Toolbar 409
Show tasks in separate windows 272
Show Types command 399
Show variable values in source code 272
Show Window Toolbar command 409
Signed Decimal command 400
single-class browser 217

single-class hierarchy, in browser 221
Smaller Code Size 314
Solaris 16, 26
    Copy Files… 416
    File Systems… 416
    Info Menu 415
    keyboard conventions 21
    Keyboard Preferences… 416
    system requirements 30
Sort Function Pop-Up 264
Sort functions by method name in browser 272
source code control 359–365
    advanced operations 366
    files 359–372
Source Code Disclosure Triangle 340
Source Code Pane 340
source files
    compiling 321
    precompiling 329–336
source pane, in browser 215
Sources checkbox 170
SourceSafe 354
Stack command 401
Start ToolServer command 411, 517
stationery 56
Step Into command 396
Step Out command 396
Step Over command 396
Stepping Buttons 339
Stop at beginning of 'main' when launching applications 276
Stop at EOF option 169, 170, 189
Stop command 396
Stop for Debugger and DebugStr traps 277
Stop ToolServer command 411
Strength Reduction 315
Switch To Monitor command 398
Switch to MW Debugger command 377
symbol documentation
    looking up 155
symbol window, in browser 222
Synchronize Modification Dates command 393
Synchronizing modification dates 84
Syntax Coloring
    and printing 118
    preference panel 266
Syntax coloring, table of 267
System Headers checkbox 171
System Paths (Windows) 301
System Paths Pane 301
System Relative Path 304
System Requirements 30

T

Target Name option 297
Target Settings command 382
Target Settings panel 295
Targets
    dependencies 91
    documentation 24
    IDE 25
    settings 293–356
Text
    drag and drop of 146–147
Text Editing Area 131
text replace
    Replace All 179
    replacing found text 178
    selective replace 178
    single file 174–181
text replacing
    in multiple files 181–191
text search 191–200
    activating multi-file 182
    Batch search 180
    choosing file sets 186
    choosing files 183
    controlling range 176
    controlling search parameters 177
    controlling search range 189
    finding selection 173
    finding text 175
    for selection 173
    multi-file 168–172
    multiple file 181–191
    overview 161
    regular expressions 191–200
    removing file sets 190
    saving file sets 187

selected text search 172
single file 174–181
THINK Reference 159
third-party editor support 244
third-party editors 85
__throw() 398
Tile command 401
Tile Vertical command 401
Toolbar Disclosure Button 136
Toolbar Elements Window command 409
Toolbar Submenu 402, 408–410
Anchor Floating Toolbar command 410
Clear Floating Toolbar command 410
Clear Window Toolbar command 409
Hide Floating Toolbar command 409
Hide Window Toolbar command 409
Reset Floating Toolbar command 410
Reset Window Toolbar command 409
Show Floating Toolbar command 409
Show Window Toolbar command 409
Toolbar Elements Window command 409
Unanchor Floating Toolbar command 410
toolbars
adding elements 285
customizing 280–288
elements 281
floating toolbar 282
modifying 285–288
rearranging elements 286
removing elements 286
restoring default settings 287
showing and hiding 282
types 280
window toolbar 284
Toolbox Assistant, Macintosh Programmer's 158
Tools Menu 410–412
Commando command 411
Execute as ToolServer Script command 412
Insert Template command 412
Lookup Symbol command 412
Start ToolServer command 411
Stop ToolServer command 411
Tools Submenu 408–410
ToolServer 22, 528–529
installing 515
(ToolServer Tools) folder 522
ToolServer Tools submenu 523
ToolServer Worksheet command 404
Touch column 83
Treat #include as #include "…" option 300
Turbo Pascal 17
Tutorial Resources 24
typographical conventions 18

U

Unanchor Floating Toolbar command 410
Undo command 379
UNIX text files 113
Unsigned Decimal command 400
updating projects 321, 322
Use BBEdit™ Extensions (Mac OS) 247
Use External Editor (Mac OS) 245
Use Modification Date Caching option 305
Use Multiple Document Interface (Windows) 244
Use Multiple Undo option 264
Use Script Menu (Mac OS) 246
Use Third Party Debugger 307
Use Third Party Editor 244
Use ToolServer Menu (Mac OS) 246
Use Version Control option 357
User Paths (Windows) 301
User Paths Pane 301
Username field 357

V

Variable change hilite 271
Version Control Settings command 382
Version Control System (VCS)
advanced operations 366
Menu 405
Window 372
View Array command 399
View As command 399
View Memory As command 400
View Memory command 399
View Variable command 399
viewers, documentation 24
Visual SourceSafe for Mac OS 49, 354

W

Warning Button 338
Warning Messages 341
Watchpoint hilite 271
Watchpoints Window command 405
wildcard searching 191–200
Win32/x86 16, 26
Window Menu 401–405
    Breakpoints Window command 404
    Browser Catalog Window command 402
    Build Progress Window command 403
    Class Hierarchy Window command 403
    Collapse Window command 402
    Errors & Warnings Window command 403
    Expand Window command 402
    Expressions Window command 404
    Global Variables Window command 404
    Minimize Window command 402
    New Class Browser command 403
    Processes Window command 404
    Project Inspector command 403
    Registers Windows command 405
    Restore Window command 402
    Save Default Window command 402
    Show Catalog Window command 402, 403
    Stack command 401
    Tile command 401
    Tile Vertical command 401
    Toolbar submenu 402
    ToolServer Worksheet command 404
    Watchpoints Window command 405
    Zoom Window command 401
Window Position and Size 263
Window Toolbar 284
Windowing panel, Debugger 272–275
Windows
    Help Menu 407
Windows 95 16, 30
Windows Help 407
Windows NT 16, 30

X

x86 Settings panel, Debugger 279

Z

Zoom Window command 401
Zoom Windows To Full Screen (Mac OS) 245

CodeWarrior

IDE User Guide

Credits

writing lead: Derek Saldana

other writers: David Blache, L. Frank Turovich, Roopa Malavally, Alisa Dean, Marc Paquette, Chris Magnuson, Jim Trudeau

engineering: Kevin Bell, Matt Fassioto, Matt Henderson, Glenn Meter, Dan Podwall, Nalini Prakash, Rob Vaterlaus, Chris Wiebe

frontline warriors: Ron Liechty, Jim Trudeau, and CodeWarrior users everywhere

Guide to CodeWarrior Documentation

CodeWarrior documentation is modular, like the underlying tools. There are manuals for the core tools, languages, libraries, and targets. The exact documentation provided with any CodeWarrior product is tailored to the tools included with the product. Your product will not have every manual listed here. However, you will probably have additional manuals (not listed here) for utilities or other software specific to your product.

Core Documentation	
IDE User Guide	How to use the CodeWarrior IDE
Debugger User Guide	How to use the CodeWarrior debugger
CodeWarrior Core Tutorials	Step-by-step introduction to IDE components
Language/Compiler Documentation	
C Compilers Reference	Information on the C/C++ front-end compiler
Pascal Compilers Reference	Information on the Pascal front-end compiler
Error Reference	Comprehensive list of compiler/linker error messages, with many fixes
Pascal Language Reference	The Metrowerks implementation of ANS Pascal
Assembler Guide	Stand-alone assembler syntax
Command-Line Tools Reference	Command-line options for Mac OS and Be compilers
Plugin API Manual	The CodeWarrior plugin compiler/linker API
Library Documentation	
MSL C Reference	Function reference for the Metrowerks ANSI standard C library
MSL C++ Reference	Function reference for the Metrowerks ANSI standard C++ library
Pascal Library Reference	Function reference for the Metrowerks ANS Pascal library
MFC Reference	Reference for the Microsoft Foundation Classes for Win32
Win32 SDK Reference	Microsoft's Reference for the Win32 API
The PowerPlant Book	Introductory guide to the Metrowerks application framework for Mac OS
PowerPlant Advanced Topics	Advanced topics in PowerPlant programming for Mac OS
Targeting Manuals	
Targeting Java VM	How to use CodeWarrior to program for the Java Virtual Machine
Targeting Mac OS	How to use CodeWarrior to program for Mac OS
Targeting MIPS	How to use CodeWarrior to program for MIPS embedded processors
Targeting NEC V810/830	How to use CodeWarrior to program for NEC V810/830 processors
Targeting Net Yaroze	How to use CodeWarrior to program for Net Yaroze game console
Targeting Nucleus	How to use CodeWarrior to program for the Nucleus RTOS
Targeting OS-9	How to use CodeWarrior to program for the OS-9 RTOS
Targeting Palm OS	How to use CodeWarrior to program for PalmPilot
Targeting PlayStation OS	How to use CodeWarrior to program for the PlayStation game console
Targeting PowerPC Embedded Systems	How to use CodeWarrior to program for PPC embedded processors
Targeting VxWorks	How to use CodeWarrior to program for the VxWorks RTOS
Targeting Win32	How to use CodeWarrior to program for Windows